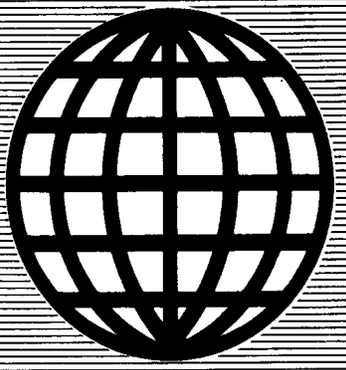


The COMPUTER JOURNAL

Programming - User Support

Applications

Issue Number 60 March/April 1993 US$4.00

10th Year Anniversary!

Four for Forth

Z-System Corner

Dr. S-100

Debugging Forth

Real Computing

Kaypro II to IV

Center Fold

Support Groups for the Classics

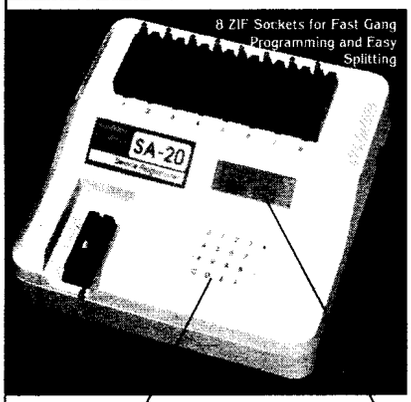
Moving Forth Part II

The Computer Corner

ISSN # 0748-9331

EPROM PROGRAMMERS

Stand-Alone Gang Programmer $750.00



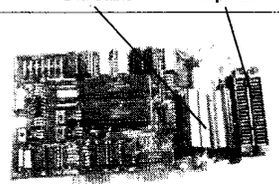
- Completely stand-alone or PC driven
- Programs E(E)PROMs
- ***1 Megabit of DRAM***
- ***User upgradable to 32 Megabit***
- ***.3/.6" ZIF socket, RS-232, Parallel In and Out***
- 32K internal Flash EEPROM for easy firmware upgrades
- ***Quick Pulse Algorithm (27256 in 5 sec, 1 Megabit in 17 sec.)***
- 2 year warranty
- Made in U.S.A.
- Technical support by phone
- Complete manual and schematic
- ***Single Socket Programmer also available. $550.00***
- Split and Shuffle 16 & 32 bit
- 100 User Definable Macros, 10 User Definable Configurations
- Intelligent Identifier
- Binary, Intel Hex, and Motorola S

20 Key Tactile Keypad (not membrane) **20 x 4 Line LCD Display**

Internal Programmer for PC $139.95

New Intelligent Averaging Algorithm. Programs 64A in 10 sec., 256 in 1 min., 1 Meg (27010, 011) in 2 min. 45 sec., 2 Meg (27C2001) in 5 min. Internal card with external 40 pin ZIF.

2 ft. Cable 40 pin ZIF



- Reads, verifies, and programs 2716, 32, 32A, 64, 64A, 128, 128A, 256, 512, 513, 010, 011, 301, 27C2001, MCM 68764, 2532
- ***Automatically sets programming voltage***
- Load and save buffer to disk
- Binary, Intel Hex, and Motorola S formats
- ***Upgradable to 32 Meg EPROMs***
- ***No personality modules required***
- 1 year warranty • 10 day money back guarantee
- Adapters available for 8748, 49, 51, 751, 52, 55, TMS 7742, 27210, 57C1024, and memory cards
- Made in U.S.A.

NEEDHAM'S ELECTRONICS
4539 Orange Grove Ave. • Sacramento, CA 95841
Mon. - Fri. 8am - 5pm PST

C.O.D.

Call for more information
(916) 924-8037
FAX (916) 972-9960

Cross-Assemblers as low as $50.00
Simulators as low as $100.00
Cross-Disassemblers as low as $100.00
Developer Packages
as low as $200.00(a $50.00 Savings)

A New Project
Our line of macro Cross-assemblers are easy to use and full featured, including conditional assembly and unlimited include files.

Get It To Market--FAST
Don't wait until the hardware is finished to debug your software. Our Simulators can test your program logic before the hardware is built.

No Source!
A minor glitch has shown up in the firmware, and you can't find the original source program. Our line of disassemblers can help you re-create the original assembly language source.

Set To Go
Buy our developer package and the next time your boss says "Get to work.", you'll be ready for anything.

Quality Solutions
PseudoCorp has been providing quality solutions for microprocessor problems since 1985.

BROAD RANGE OF SUPPORT
- Currently we support the following microprocessor families (with more in development):

Intel 8048	RCA 1802,05	Intel 8051	Intel 8096
Motorola 6800	Motorola 6801	Motorola 68HC11	Motorola 6805
Hitachi 6301	Motorola 6809	MOS Tech 6502	WDC 65C02
Rockwell 65C02	Intel 8080,85	Zilog Z80	NSC 800
Hitachi HD64180	Motorola 68000,8	Motorola 68010	Intel 80C196

- All products require an IBM PC or compatible.

So What Are You Waiting For? Call us:

PseudoCorp
Professional Development Products Group
716 Thimble Shoals Blvd, Suite E
Newport News, VA 23606
(804) 873-1947 FAX: (804)873-2154

FORTH

Journey with us to discover the shortest path between programming problems and efficient solutions.

The Forth programming language is a model of simplicity: In about 16K, it can offer a complete development system in terms of compiler, editor, and assembler, as well as an interpretive mode to enhance debugging, profiling, and tracing.

As an "open" language, Forth lets you build new control-flow structures, and other compiler-oriented extensions that closed languages do not.

Forth Dimensions is the magazine to help you along this journey. It is one of the benefits you receive as a member of the non-profit Forth Interest Group (FIG). Local chapters, the GEnie™ Forth RoundTable, and annual FORML conferences are also supported by FIG. To receive a mail-order catalog of Forth literature and disks, call 510-89-FORTH or write to:
Forth Interest Group, P.O. Box 2154, Oakland, CA 94621.
Membership dues begin at $40 for the U.S.A. and Canada.
Student rates begin at $18 (with valid student I.D.).

GEnie is a trademark of General Electric.

SAGE MICROSYSTEMS EAST

Selling and Supporting the Best in 8-Bit Software

Z3PLUS or NZCOM (now only $49 each)
XBIOS for SB180 ($50)
ZMATE text editor ($50)
BDS C for Z-system (only $60)
DSD: Dynamic Screen Debugger ($50)
PCED: ARUNZ and LSH for MSDOS ($50)
ZMAC macro-assembler ($50, $70 with printed manual)
Order by phone, mail, or modem and use
Check, VISA, or MasterCard.

Z-System public domain software by mail.

Regular Subscription Service
Z3COM Package of over 1.5 MB of COM files
Z3HELP Package with over 1.3 MB of online documentation
Z-SUS Programmers Pack, 8 disks full
Z-SUS Word Processing Toolkit
And More!

For catalog on disk, send $2.00 ($4.00 outside North America) and your computer format to:

Sage Microsystems East
1435 Centre Street
Newton Centre MA 02159-2469
(617) 965-3552 (voice 9 to 11AM)
(617) 965-7259 (pw=DDT)
(MABOS on PC-Pursuit)

The Computer Journal

Founder
Art Carlson

Editor/Publisher
Bill D. Kibler

Technical Consultant
Chris McEwen

Contributing Editors
Herb Johnson
Charles Stafford
Brad Rodriquez
Matt Mercaldo
Tim McDonough
Frank Sergeant
Clem Pepper
Richard Rodman
Jay Sage

The Computer Journal is published six times a year and mailed from *The Computer Journal*, P. O. Box 535, Lincoln, CA 95648, (916) 645-1670.

Opinions expressed in *The Computer Journal* are those of the respective authors and do not necessarily reflect those of the editorial staff or publisher.

Entire contents copyright © 1992 by *The Computer Journal* and respective authors. All rights reserved. Reproduction in any form prohibited without express written permission of the publisher.

Subscription rates within the US: $24 one year (6 issues), $44 two years (12 issues). All funds must be in U.S. dollars drawn on a U.S. bank.

Send subscription, renewals, address changes, or advertising inquires to: *The Computer Journal*, P.O. Box 535, Lincoln,CA 95648.

Registered Trademarks

It is easy to get in the habit of using company trademarks as generic terms, but these trademarks are the property of the respective companies. It is important to acknowledge these trademarks as their property to avoid their losing the rights and the term becoming public property. The following frequently used trademarks are acknowledged, and we apologize for any we have overlooked.

Apple II, II+, IIc, IIe, Lisa, Macintosh, ProDos; Apple Computer Company. CP/M, DDT, ASM, STAT, PIP; Digital Research. DateStamper, BackGrounder ii, Dos Disk; Plu*Perfect Systems. Clipper, Nantucket; Nantucket, Inc. dBase, dBASE II, dBASE III, dBASE III Plus, dBASE IV; Ashton-Tate, Inc. MBASIC, MS-DOS, Windows, Word; MicroSoft. WordStar; MicroPro International. IBM-PC, XT, and AT, PC-DOS; IBM Corporation. Z80, Z280; Zilog Corporation. Turbo Pascal, Turbo C, Paradox; Borland International. HD64180; Hitachi America, Ltd. SB180; Micromint, Inc.

Where these and other terms are used in The Computer Journal, they are acknowledged to be the property of the respective companies even if not specifically acknowledged in each occurrence.

TCJ The Computer Journal

Issue Number 60 March/April 1993

Editor's Comments 2

Reader to Reader 3

Next Ten Years Part II 8
The last word on TCJ's future.
By Bill Kibler.

Support Groups for the Classics 10
A new feature on helping you find support..
By J. Wm. Weaver.

Z-Systems Corner 11
Reviewing the past 8 years of Z-system.
By Jay Sage.

DR. S-100 15
Letters and collecting computers.
By Herb R. Johnson.

Four for Forth 19
A review of Forth engines.
By Jack Woehr.

Real Computing 23
Reviewing UNIX, Minix, and Coherent.
By Rick Rodman.

Center Fold 25
IMSAI CPA, the Front Panel.

Debugging Forth 30
Some good advice and tips on working with Forth.
By Walter J. Rottenkolber.

Mr. Kaypro 36
Making a Kaypro II into a IV.
By Charles B. Stafford.

Moving Forth 40
Part II, case studies of Forth kernels on different CPU's.
By Brad Rodriguez.

The Computer Corner 51
Reviewing good books on computing.
By Bill Kibler.

EDITOR'S COMMENTS

Welcome to the Tenth Year of The Computer Journal. This is our 60th issue of TCJ and with that we want to thank all our readers for supporting us for all these years.

In this issues we have a special letter from Art Carlson, the founder of *The Computer Journal.* Art asks what projects interest you and how he might be able to help. So please read his comments and drop him a letter to show him how much we miss seeing his words in print.You will find his reflections on the next page, our Reader-to-Reader section.

Walter J. Rottenkolber provides his special review of the past after Art's. Walter also provides excellent support for beginning Forth users on page 30 as he explains how to debug Forth (also some good advice for any language - like Keep It Simple). Walter's letter shows just how diverse our writers and readers interests and backgound in computers can be. In fact Walter owns a system I haven't hear about before. Helmut Jungkunz follows Walters letters with some comments and ARUNZ scripts.

Our regular writers have incorporated their thoughts on the past and future within their articles. Jay Sage takes this issue to review his past eight years of articles by commenting on the development of the Zsystem as he and others explained it in *TCJ* columns You will be amazed just how much we have covered in the last Ten Years, I know I was!

Rick Rodman covers UNIX, Minix, and Coherent, before commenting on why 32bit systems represent Real Computing. Our S-100 supporter, Herb Johnson, answers a letter or two and then speculates on the future of collecting classic systems. Answering letters is also done by Mr. Kaypro or Chuck Stafford. His answers show how to covert an old Kaypro II into a newer IV.

A new feature starting with this issue is "Support Groups for the Classics" by J. Wm. Weaver. JW will be attempting to help our readers find organizations or groups that support your interests in computing systems. This will not be a simple job without your support. JW will be collecting information by region (including foreign) and reporting here on what he finds. He will need your input, so please read his introduction and send him information about your favorite organization.

Our special articles start with the second edition of "The Next Ten Years." This position paper summaries your comments and feed back I have received since the first version appeared in issue #56. Let me make it clear, that this is the last installment on this topic!

Brad Rodriguez provides part two of moving Forth and presents case studies of actual Forth implementations. In issue #61, Brad provides his long awaited 6809 Multiprocessing project with the hardware layout. This is Multiple CPU's running one system, not one CPU running mulitple users. What's more, he starts with the hardware part of the project, first. I am really forward to this one. Thanks, Brad!

Mr. Jack Woehr honors us in this special issue with a review of Forth hardware engines. If you think the new 586 is suppose to be a fast computer, you need to read Jack's article and see what speed really is about. The article reviews those high speed devices presented at the last Rochester Forth Conference.

The last word from me, is a review of books to get and use. I have had many requests from reader as to what books to get and I have answered those requests in my Computer Corner. Due to the size of this issue, I have delayed the start of a new operating system support column. Since many of our new readers have little experience with CP/M and other eight bit operating systems, I am starting a new series on operating systems. The first set of articles will cover the basics of CP/M. Later series will cover other systems such as Flex, and OS9.

All in all this issue provides reviews, new items, and great topics for your reading enjoyment and pleasure.

Please note the special offers inside. For our advertisers, I am rolling back the cost to 1982 levels, or HALF PRICE! That is right, 1982 levels, so order two advertisements and only pay for one. Like all good offers this only applies to the next three issues.

One offer with no time limits is the 10 and 15% discounts on large orders of back issues. See page 49 for details.

READER to READER

Letters to the Editor

All Readers

MINI Articles

SPECIAL LETTER FROM ART CARLSON

The Birth of TCJ

It was ten years ago that I bought my Apple II+ and started searching for the information I needed to make it do more than just run the few available canned application programs. What we had to work with then is almost comical when compared with what is available today. My Apple was fully stuffed with a whole 64K, and the display was limited to 40 characters of upper case, but it did have a crude graphics mode. The single sided floppy drives held what at that time was considered a gigantic 135K -- people laughed at me because I bought a whole box of ten disks (priced at $4.50 per disk) because no one could ever fill up the entire ten disks. Now at work I use a 33 MHz '386 with 8 Meg of RAM and a 512 Meg hard dive, and buy 1.44 Meg disks 50 at a time. But you know what? Today's computers are no longer any fun!

The Apple was a good learning tool at the time. With its built-in BASIC and Monitor plus a S-C assembler I peeked and poked, and started to learn programming. The addition of a John Bell A/D (Analog to Digital) card whetted my interest in interfacing to real world devices instead of just writing another ball-bouncing BASIC program. That's when I really got hooked, and my life has never been the same since.

I soon bumped into the Apple's limitations, and got a REAL computer with a 4MHz Z-80 that ran CP/M (which I soon upgraded to ZCPR). It still had only 64K, but it had a real operating system and a TV-950 terminal with 80 columns plus upper and lower case. I still remember Donna asking if I really needed two computers -- after all one should be enough for any sane person. Now, she just admits that I have lost my sanity, and sighs when I bring home ANOTHER computer that I have saved from the dump. I regret that I recently missed a complete North Star system that went to the land fill because nobody was interested in the obsolete relic. Oh well, there is always another chance on something else tomorrow, and this way I can still (barely) get the car in the garage. We have 27 acres and I have considered building another garage, but the are limits to how much I can pile up. Does anyone know of a home for two Teletype machines? I've already scrapped out two other ones for parts.

After becoming familiar with the CP/M system I had even more questions about interfacing and control. I was running a print shop at the time, and I kept complaining that the available books and magazines didn't have the information that I needed. So, after several friends suggested that the only way that I could get the information was to start a magazine, I started TCJ. Now, there was a lot of information available, but I was so busy editing and publishing that I didn't have time to play with the computers. I'm ashamed to admit that I recently dusted off a six year old list of projects which were never completed. But my real interests haven't changed, and now that I am moving into semi-retirement I still intend to work on them.

Moving Onward

I am now moving into semi-retirement, and will have more time to work on hardware/software projects. I have the original Apple II+, five CP/M systems, two MS-DOS systems, plus buckets full of chips and parts. For equipment I have a Tektronix dual beam scope, a Needham EPROM burner, plus a EPROM eraser. Most of my work will be in assembly, so the PseudoCorp cross-assemblers and simulators will take care of the language needs.

I intend to concentrate on small controllers based on the Z80 and 8051 families, which include the Z8 series, the Z180 (HD6480), 8052, 80C552, etc. Many of these will be stand-alone units, some will be multi-processor systems which communicate with each other, and a few will communicate with a computer motherboard.

I fully agree with Bill that the project modules should interface over a serial link when ever possible. One of the primary reasons for this choice is that we open the projects to everyone who has a serial port available. Other people may have projects which must connect directly to the motherboard bus in order to obtain sufficient speed for large amounts of data, but that is not the type of project which I intend to work on.

As an example of a project with a small amount of communications traffic, I would like to record the daily minimum and maximum ground temperature every six inches down to a level of four feet and upload the data to disk once a week. That is 16 bytes a day, or 112 bytes a week. Any system with a serial port can handle that very easily. I only regret that I don't have date for the past ten years, because I feel the variations in the subsoil temperature will be very useful in

determining if there really is a significant warming trend.

Most of the projects at the top of my list involve timing, measurement, and motion control. And most of them must be small, portable, low power, and low cost, which rules out the use of a computer.

Where Do We Go From Here?

I am not sure how many reader's interests coincide with mine. I'll be designing small project boards, evaluating components, wirewrapping prototypes, writing assembly language programs, and wrestling with the trade-offs in real world control. Almost all of my project software will be self-contained, and will not use an operating system such as CP/M or MS-DOS. The serial link will be used for data and/or commands where extensive data manipulation or graphic displays are required.

I have been forced to start using a powered wheelchair, and am not satisfied with anything available, so I'll design my own. Much of my work will be in the areas of measurement, motors, motion control, and robotics.

I would be interested in helping to establish a disk library of assembly language routines related to measurement and control. I'm also interested in preparing a tutorial on the selection and application of motors for motion control.

Should TCJ include this type of material? Tell Bill how you feel about this, or contact me at:

Art Carlson
190 Sullivan Crossroad
Columbia Falls, MT 59912
(406) 257-9119 (Voice--6 to 9 p.m. or weekends)

Thanks for the Article/Letter Art. I know that all our readers would like to personally thank you for starting TCJ! However, if they all did, your mail person would be most unhappy, but then what better way to say thanks than sending you a personal note.

We all are looking forward to those articles you talk about, and I will find space as soon as you send them to me. TCJ has only gotten better over the years thanks mostly to your good start. Now all we have to do is let others know about what you did in the past and what we are doing for the future.

Thanks again Art, Bill Kibler and all of TCJ's readers.

Dear Mr. Kibler,

First, I'm enclosing a short article on some words that turns Forth into a powerful script language for a Forth modem program. As you can see, it doesn't take many words.

Second, about the last ten years of computing.

Ten years ago I bought my first and still primary computer system, a Kaypro II. Back then, computers had made the move from 'hobby' to 'home' computer status, and my thought was to jump on the computer train before it passed me by. At the time, I could discover only two really good reasons for spending the then pricely sum of $1,800.00, word processing and learning about computers. These reasons still hold true, except that I've combined them by writing articles about my computer tinkering. The Kaypro, of course, hasn't stayed the same. Micro Cornucopia character and Pro-8 ROM upgrades, larger disk drives, a 5 Mhz speed change, and a Micro-Sphere 1MB Ram Disk make computing much more pleasant.

I also bought an Atari 800XL, an Imsai-8080, a SWTp 6809 and a Data-Master 32 (the latter two not working, alas). This has sparked an interest in history and development of computers.

These simple systems can't be beat for learning about inner workings of a computer, especially if you aren't a Wizard. The 8080/Z80 cpu's are a paradigm for register based systems, while the 6502/6809 cpu's represent the memory based systems. A great body of technical information on hardware, and source code for programs and operating systems reveal the mysteries of these simpler computers.

Times change. At a recent computer conference, Bill Gates and Phillipe Kahn spoke of the future. Mainframes on a desk capable of 100 MIPS, with 32 MB RAM (on up) and gigabyte hard drives, running programs based on 10 million lines of code. Programming will be by linking standardized black-box objects together. These objects would be created by third party groups specializing in different aspects of a program (lans, graphics, etc.). The language of choice would be C++, though any language capable of producing the object to standard could be used.

As a first step, Borland released its new version of C++, which it is using to develop all its new product releases. It took some 120 man years to create, weighs in at 30 pounds and takes 27 MB of hard drive space (and 5 more to install). Are we talking about mere mortals here? NOT! Even professional programmers admitted to being at sea with sharks circling.

So long life and prosper to 'The Computer Journal'. Perhaps it can lead the way back to the future when computers were on a human scale and programs had less than 200 pages of documentation.

Yours truly,
Walter J. Rottenkolber
Mariposa, CA.

Well Walter your letter is just perfect and expresses my ideas perfectly. I have one of those C++ book collections at work and can only put it on a lower shelf for fear the shelf will break and kill someone. I am lucky in that I only work on 68000 assembly and don't do much C++. I often see those same books take flight when one of the unfornutnate programmers gets so frustrated trying to figure out what they are suppose to do that they toss the book (would throw the computer if it wasn't so heavy) at the nearest trash can.

I am running a bit short of space this issue and will run your "script" article

latter. We do need to see about helping you get your other two systems running. I know you are interested in getting the SWTp started up again, especially now that Brad Rodriguez is showing us how to do a 6809 Forth system. Maybe these 6809 units would be a good candidate for starting on my universal operating system (since I have a GIMIX that runs, but needs more software). By the way, what is a Data-Master 23, it is one system that I don't seem to recall.

I appreciate your words and especially your articles. Thanks for being a TCJ writer! Bill Kibler.

Various matters concerning TCJ:

Dear Bill,

I am an eager reader of TCJ and I find some of the articles very interesting, but in some respect, I find TCJ to have changed. I read your discussion with Tilmann Reh about specialist themes and so on, but I think, the real situation boils down to two main aspects:

The few specialist articles that deal with specialist ideas or programs and the mass of people who are interested in getting a possibly global view of "the scene" of the 8-Bit world.

I personally feel a bit left out, if I compare the interesting articles on the development of the general ZCPR-tools or CP/M Public Domain to the High-Tech articles of today that are maybe interesting as high-lights, but not truely satisfactory for the "standard computer user". I also think of TCJ as a valuable instrument to maintain programmers' contacts, but it shouldn't be forgot, that the average computer user is a more or less naive person, hacking in his own living room world and trying to understand, what he (or she) is doing and why it does (or doesn't) work.

Therefore I think it essential to explain programs and concepts in a simple language (not for idiots, but legible for strangers like us sour krouts here). There are a lot of things to be said and to be done of which the average user has much advantage, like tricks of the trade. A very good example of what I mean were early articles in M.O.R. and the F.O.G.Horn, such as "Tools for Tyros" and the great ZCPR-articles by Rick Charnes "Forever Z", that sent me on my way to Z-World. Also, I'd like to say, that nothing can hurt a technically oriented magazine more, than if the publishers and writers take themselves too seriously. Pieces of Eight (Eight bits & Change) showed, that humor has a very good right to appear in a techno-magazine. After all, we're all people, aren't we?

This is why I sit down every now and then to think over the many little problems I get presented every day by people calling me up or writing letters, most of them dealing with typical CP/M problems (or ZCPR ones). Here is one thing that obviously nobody dealt with before, a really small thing, some people might think, but a big problem for others. What I'm talking about is, how to write an automatic letter with ZDE with CP/M Plus, which carries built-in date and time anyway. Remember, there was a solution with DW.COM, a program that would tell you the Day of Week and program a ZCPR-register. Another file would then be processed through complicated SH-scripts and ZEX to redirect the input to ZDE, my favourite Editor.

Since DW only works with CP/M 2.2, it couldn't be used for CP/M Plus. So, while thinking it through, I suddenly remembered the variables used by Jay Sage's fantastic ARUNZ.COM. The first step was to look at an alias called DAYTIME, that I found on one of the disks Jay sent me:

```
DAYTIME         echo    ^M^JD%>ate:
$DD.$DM.$DY^J^M%<Time:
$DH:$DC:$DN^J^M
```

this would output the folling:

```
Date: 27.02.93
Time: 00:12:03
```

Then I reviewed the DATLET.LBR and found the two aliases to work best:

```
PUTDATE  REG s3 $dd;go s2 $dm;go s4 $dy;go
s5 $dh;go s6 $dc;go s7 $dn
DL2      $zfi;ZEX DATELET $1 $2 $rf03 $dy
$rf04 $rf05 $dn $da
```

The final alias starts my standard letter:

```
LETTER  PUTDATE;DL2 ZDE $1
```

What actually happens, is that the CP/M Plus date and time is broken down to small ARUNZ-variables and successivly poked into the ZCPR (Z3PLUS) registers via the tool REG.COM. Then the file, whose name had been passed to ZDE as parameter, has to be opened by ZEX and the scriptfile DATELET.ZEX automatically substitutes the name of the month corresponding to the contents of register 2, so that the entry becomes easier to read. The only disadvantage of the method used, is that you have to copy your own address twelve times, once for each month and to pass the right parameters to it. The ZEX commands FALSECMD and ZEXMSG OFF followed by QUIET shut off the talkative output of ZEX running.

To add cosmetics, you can add tabs and spaces into your address lines. It should also be mentioned, that you have to use IF.COM in order to check the contents of your registers. To find out, when IF.COM is necessary, I renamed IF.COM to IFCOM.COM. That way, I find it easier to use the right IF version.

This may seem a very small thing for some people, especially for those, who consider themselves cracks, but "Real programmers don't use Pascal .." and so on. It was a problem I wanted to solve and I did, and that is what makes computing interesting for me. It is the main reason, I still stick to the 8 bits, because things are understandable and happen in a small box, where I can lay my silly hands on myself without getting hurt or hurting anybody. Shucks, I like to relax while hacking about!

With these thoughts I'd like to encourage all the "movers in small paces" to come out with their small solutions and everyday wonders. They should understand that there is always someone who gets new ideas out of theirs and that they should send the spiral spinning. No

small things - no big things! I am not the super-shiny brain in Z80-programming, but I know I have initiated more good programs than many others and given very good ideas to lots of friends , simply because I believe in communication.

By the way, I'm hunting for all sorts of C-128 tools and programs. Why? Because the C-128 is one of the least supported computers here in Germany, all of a sudden. Commodore not even care to supply people with CP/M Plus system disks, so many have to get copies of system disks wherever they can. If you want to help with your ideas you can send mail to me via compuserve 100024,1545 (Helmut Jungkunz). I'd be glad to answer as soon as possible.

```
---------- modified -------- DATELET.ZEX -- for
CP/M Plus ----------
|FALSECMD|
|ZEXMSG OFF|
|QUIET|
ifcom 2 1
$1 $2
<^I^I^I^I^I^IHelmut Jungkunz
<^I^I^I^I^I^IZacherlstr. 14
<^I^I^I^I^I^IW-8045 Ismaning
<
<^I^I^I^I^I          Ismaning, $3. Januar 19$4
<^Qc
else
ifcom 2 2
$1 $2
<^I^I^I^I^I^IHelmut Jungkunz
<^I^I^I^I^I^IZacherlstr. 14
<^I^I^I^I^I^IW-8045 Ismaning
<
<^I^I^I^I^I          Ismaning, $3. Februar 19$4
<^Qc
else
ifcom 2 3
$1 $2
<^I^I^I^I^I^IHelmut Jungkunz
<^I^I^I^I^I^IZacherlstr. 14
<^I^I^I^I^I^IW-8045 Ismaning
<
<^I^I^I^I^I          Ismaning, $3. M{rzr 19$4
<^Qc
else
ifcom 2 4
$1 $2
<^I^I^I^I^I^IHelmut Jungkunz
<^I^I^I^I^I^IZacherlstr. 14
<^I^I^I^I^I^IW-8045 Ismaning
<
<^I^I^I^I^I          Ismaning, $3. April 19$4
<^Qc
else
ifcom 2 5
$1 $2
<^I^I^I^I^I^IHelmut Jungkunz
<^I^I^I^I^I^IZacherlstr. 14
<^I^I^I^I^I^IW-8045 Ismaning
<
<^I^I^I^I^I          Ismaning, $3. Mai 19$4
<^Qc
else
ifcom 2 6
$1 $2
<^I^I^I^I^I^IHelmut Jungkunz
<^I^I^I^I^I^IZacherlstr. 14
<^I^I^I^I^I^IW-8045 Ismaning
<
<^I^I^I^I^I          Ismaning, $3. Juni 19$4
<^Qc
fi
fi
fi
fi
fi
fi
ifcom 2 7
$1 $2
<^I^I^I^I^I^IHelmut Jungkunz
<^I^I^I^I^I^IZacherlstr. 14
<^I^I^I^I^I^IW-8045 Ismaning
<
<^I^I^I^I^I          Ismaning, $3. Juli 19$4
<^Qc
else
ifcom 2 8
$1 $2
<^I^I^I^I^I^IHelmut Jungkunz
<^I^I^I^I^I^IZacherlstr. 14
<^I^I^I^I^I^IW-8045 Ismaning
<
<^I^I^I^I^I          Ismaning, $3. August 19$4
<^Qc
else
ifcom 2 9
$1 $2
<^I^I^I^I^I^IHelmut Jungkunz
<^I^I^I^I^I^IZacherlstr. 14
<^I^I^I^I^I^IW-8045 Ismaning
<
<^I^I^I^I^I          Ismaning, $3. September 19$4
<^Qc
else
ifcom 2 10
$1 $2
<^I^I^I^I^I^IHelmut Jungkunz
<^I^I^I^I^I^IZacherlstr. 14
<^I^I^I^I^I^IW-8045 Ismaning
<
<^I^I^I^I^I          Ismaning, $3. Oktober 19$4
<^Qc
else
ifcom 2 11
$1 $2
<^I^I^I^I^I^IHelmut Jungkunz
<^I^I^I^I^I^IZacherlstr. 14
<^I^I^I^I^I^IW-8045 Ismaning
<
<^I^I^I^I^I          Ismaning, $3. November 19$4
<^Qc
else
ifcom 2 12
$1 $2
<^I^I^I^I^I^IHelmut Jungkunz
<^I^I^I^I^I^IZacherlstr. 14
<^I^I^I^I^I^IW-8045 Ismaning
<
<^I^I^I^I^I          Ismaning, $3. Dezember 19$4
<^Qc
fi
fi
fi
fi
fi
fi
---------- remember - this is the European
version! -----
```

Where you see <^Qc you must use ZDE to enter a real ^Q after the < to do the trick (^P^Q). What it does, it sends the cursor to the end of the file, at this time right after your address header!

regards
Helmut Jungkunz
Germany

Thanks for sending me this message VIA Compuserve. Well you covered lots of ground in comments and thanks for the ZCPR tips. Getting our writers to be less "techno" has been a real problem. I think we are starting to get a handle on it, but expect it to take much more time. I have noticed that one magazine has taken our complaints to heart and has added more beginner articles. I helped show one prospective writer how to enhance their article with a more personal and explanative approach and it is now in that other magazine.

One main problem for the present is we don't pay for articles. I hope to change this soon, but until I do, many writers will go where there is money to be made. That means we get good articles, but can not solicit a know writer who has the special skill it takes to explain something in ways that all readers can understand. I am working on this, but just don't have the time it really takes.

I believe that many of the MicroC Kaypro disks are C128 compatible but of course not CP/M 3 utilities. Maybe some of our readers can let me (or you) know of a good source for what you need.

Thanks again for the comments, Bill.

Dear Mr. Kibler,

I tried calling but you were on vaction. I don't know about rate increases, but rather than wait for the minute I am enclosing a check for $32 for two years renewal.

It is great to see Kayrpo support. Two of my machines already have the relocated reset button. I am in the process of upgrading some Kaypros with Advent hardware from Stafford and NZCOM from Sage. There is alot of stuff on the SIMTEL20 mirrors, so I am wondering how useful the Micro Cornucopia disks will be. Can you also get schematics? I have one for the K10, but understand that there are others. I am looing forward to seeing the speed and power supply upgrades.

I also have an Intersystems S-100 system I would like ot get back up. I have my eye on a similar working system which I might be able to use to diagnose mine. Will anyone rework the IDE controller for S-100; what needs to be put in the BIOS? I've implemented the IOBYTE, the public drive patch and installed an Electrologics Q-Disk (RAM disk), and therefore not quite as afraid to mess up my system if it ever works again.

Yours truly,
Preston Bricker

Thanks Preston for the renewal and comments. I am aware of the limited response on the Kaypro Disk. I have only sold 2 disks to date. I know very little about SIMTEL20, but maybe our new writer (JW Weaver) can comment on it later for us. We do have some schematics and I will be listing which ones later.

I am still trying to get an article out of anybody that will explain the insides and how to program to IDE drives. However they all says it is so simple no one seems to want to write about it. After reading some comments on compuserve, I can tell it is not very simple at all. It is starting to look like I will have to do it myself, and well that is just going to take some time (currently having trouble finding specifications on the actual interface standard - if there is one). As for using IDE on S-100, once we get a specification on it, I will be able then to say what needs to be done.

Thanks and let us know how the S-100 repair job goes, Bill Kibler.

Dear Sir:

I am currently receiving your trail subscritptions. Enclosed is a check for $44.00 for a two year subscription. My thanks go to Frank Sergeant for recommending your magazine.

My current interests include Forth, embedded systems, the 8051 and 68HC11. From what I've seen of your magazine, it tends to provide more information on software side of things. I would like to see alittle more on hardware as well as gutsy type topics on low level hardware/ software interfacing and processor/ protoypt start-up. There never has been much published and it's becoming something of a lost art.

In general your magazine fills a niche that has been empty for far too long. Keep up the good work.

Sincerely yours,
Mark. E. Bender

Thanks Mark. We are trying to change the amount of hardware to software in TCJ. Next issue has a 6809 project by Brad Rodriguez that helps tie all his Forth and assembler articles together. I have added the center fold to help hardware types find lost schematics. We are getting a more even balance, but again, change takes time. I have had other articles promised, but the writers have either found paying users, or just are too busy to finish what they started for us. Thanks again, Bill.

Gentlemen:

I'm returning my collection of TCJ's to you under separate cover. As you cna see, I've maintained my subscription for a considerable time--always hoping that a KEY would emerge which would begin to make sense out of it all. Ypu may be able to find a better home for the back issues.

My letter to your predecessort appeared in Issue 53. The response was not particularly reassuring. Let me quote that response: "TCJ was never meant to be light reading."

Your aimes as stated in Issue 56 seemed cordial enough--perhaps SOMEBODY would fdinally attempt to anglicize an article here and there and I'd find that elusive key. Alas! It hasn't happened. Never a sentence to be seen which lacks some cryptic abbrevation or mnemonic familiar to every insider but a guaranteed obfuscator for those of us that tend to communicate in a spoken language.

I offer my sincere pity to your contributors. They'll undrego a complete change of attitude before they ever experience the real high associated with providing some willing and deserving soul with a clear answer to a legitimate question.

Yours very truly,
Earl Bryant.

Well Earl, I was sadden to see the back issues, but it was not a surprise in retrospect. I have had others drop TCJ for just the reason you mentioned. I must say I am working on the problem, but changing the habits of many years is no easy problem to overcome. Yes we have been heavy reading, but that is no excuse for not providing the resources to lighten and expand the knowledge that people like yourself need to understand the subject. As a past part time Junior College teacher, I saw many levels of students. Unfornately there are always a few who never get it. One past instructor would wash out 60% of his students using that premiss. I took the same course and changed it to less than 20% didn't make it. So yes I see your position and am working to change it. It takes time however.

I'll keep your collection sitting on my desk just to remind me that some of our readers don't want to be forgotten. BDK.

60th Feature

Editorial Comment

Updated Future

Next Ten Years Part II

By Bill Kibler

In issue 56 I laid out some thoughts on where *TCJ* would be going over the next TEN YEARS. Since that issue, many readers have sent in their comments, our writers have had a chance to consider the options, and I have pondered the topic endlessly. What has distilled out of those discussions and comments has been minor changes in our direction and a clarification of new options and possible directions to follow. What I will try and relate here, is those options and how they might effect you, our readers.

The Options Before Us.

A number of options drifted to the surface over the past few months. A major driving force of those options has been the economy. The results has been to make sure the readers dollar gets them something they can use right now. We have seen our readership drop, not from any changes we did, but simply because so many readers have lost their jobs and have been forced to do without items once considered important.

The magazine world has not been doing well either, with several publications ceasing production. *Elektor* magazine stopped United States production because of low returns on their investment. Several smaller newsletters stopped due to shortage of readership. The impact of IBM clones continues to focus readers away from alternatives in computer systems and thus alternative publications like ours. I dare say the history of computers for many users started with IBM clones and they probably believe that nothing existed before that.

Like any magazine *The Computer Journal* must carve out it's own readership and provide something of value to their supporters. We never have been a clone magazine and have no plans on supporting the current madness of bigger and bigger is better philosophy. In fact our philosophy would be just the opposite, that smaller and simpler you can make the product or project the better and more efficient it will be.

That concept pretty much makes us a non-clone magazine and it is at that point which I can look around and see a vast market of unsupported computers. We have always supported S-100, Kaypros, rolling your own systems (like YASBEC), and general do-it-yourself projects. Yes there are other magazines doing that to, but they **require** an IBM clone systems for their projects. Often we find a clone an excellent system for our work, but at *TCJ* the desire is to give you alternatives if you chose a different direction than the rest of the crowd.

Classic Market

I have tried to see if any magazine is supporting the CLASSIC computers. I quickly picked up one magazine that had an article using "classic" computers, only to find their concept of old or classic was an XT clone (PC clones were too old even for them). So I feel that we can honestly say, that *The Computer Journal* is the only magazine that has a stated desire and interest in helping users keep their older "classic" systems running.

There are many little newsletters that support various special aspects of classic computing. I have someone that will be providing a column that comments and lists what is going on with those publications. I have had conversations with two such newsletters and they have indicated that they might stop their work in the near future. Each has told me that should they decide to do so, we can look forward to their comments appearing regularly in *TCJ*.

For most of *TCJ*'s life we have focused our attentions on CP/M based systems. Yes we have covered other systems, but without much follow through. If we are to support the unsupported classic user, it means an opening up to other products. I subscribed for many years to *68Micro*, a magazine that supported 6800 through 68000 systems. These were GIMIX, SWTP, and Radio Shack COCOs, running such operating systems as FLEX, and OS9. When the owner of 68Micro passed away, his sons and others of the corporation had no desire to continue the magazine. I have been talking with the corporation in hopes of getting their user disk collections and possible back issue rights. I expect this to take some time and can only say that *TCJ* will be supporting these items no matter what the outcome of talks with *68Micro*.

Are their other machines we could support. Sanyo MBC machines are non-clone machines. These early MSDOS based machines have a similar architecture as a clone, but typically differ enough that only the simplest of software will run. Hardware compatible they are not in any way except running an 8088 CPU. Since CP/M was intended to run in a small memory space, many of our topics can certainly be applied to these classic clone units. I think programs like FPC Forth, MYZ80, and such, provide the perfect avenue for those user of non-clone systems to enhance their product. Our hardware articles certainly can add

skills that the reader can use to make hardware changes to these limited systems.

EDUCATION

One area which I think *The Computer Journal* can excel without much competition is in education. Now this is not about educational computers, for those are becoming large expensive clones, but about affordable hardware and software tinkering machines. To learn about computers you must "tinker." What is tinkering, mostly being able to change some aspect of your system and see the results. Many of course will say that you can do that on a clone machine, and of course that is true if you don't want to change the BIOS, or disk formats, or I/O operations, and forget about changing the operating system. In short all you can learn about on a clone machine is how to put an expansion card in a slot and use "C" programs to read and write to it.

Over the years I have enjoyed and learned considerably more than is possible from just playing with simple I/O. My most educational experiences was involved in making S-100 cards talk to each other (helped me master concept of BUS data transfers, hardware interfacing problems, and design tradeoffs), adding new BIOS features (how to program at the hardware level, disk control problems and disk format considerations), changing 4K memory devices to 16 and then 64K devices (processor and memory timing problems and signal line handshaking), and hacking and more hacking to gain knowledge.

To design todays glue logic requires knowledge of past discrete logic design. Yes, todays modern designs are based on knowing how the old systems were put together. So if you want to design new and better products nothing can replace understanding and changing classic computers. Is it an expensive task, not at all! Classic computers typically can be had for $50 (bought a Kaypro 2 with all manuals and programs for $50 last week-looked like brand new as well). What happens if you do something wrong and smoke your board. With a classic system you might be out $50, but heaven help you if your using your 486 clone with a $600 motherboard and you slip and short out the main ASIC (does all the things discrete logic did.)

Classic systems are by nature simple and straight forward. All devices typically were simple TTL logic devices. Their cost are pennies and can be found on scrap systems in your computer junk collection. When making hardware changes there is little fear of "shocking" associated components, a common problem with current CMOS devices (TTL is rather static shock insensitive, where as CMOS are very static sensitive parts.) When I first did my S-100 work, I used a slow 35MHZ oscilloscope. For 10MHZ or greater troubleshooting, a 100MHZ scope is almost the minimum necessary and with it comes a $1000 to $2000 price tag. Scopes like that are not often found in beginners workshops. Can you work on high speed logic without a logic analyzer, seldom these days. On old 4MHZ Z80 systems, I repaired many a problem using my slow scope that now days (clones at 33MHZ) would require a very good analyzer. All these items make a big difference for the beginning or learning hardware hacker.

For beginning programmers, finding simple systems that minor changes will NOT crash something else is a problem. These classic computers have limited software to get in the way of your project. Many early systems have very good diagnostic ROMs that will allow you to bring up the computer without a disk operating system. If you want to learn how to talk to hardware from your programs, and not have the BIOS turning off your port when your not looking, then try classic systems. Many of the current clone BIOS's will turn off your ports in the middle of your program, not a fun experience at all. Classic systems were made with the programmer having total control, not some ROM BIOS! You can be pretty sure that if it doesn't work on a classic system, it is your own fault and not the operating system trying to do it for you.

Where Now

For the next TEN YEARS, *TCJ* will be trying to bring all the above about. What this means for our readers is better and better articles. Most of the minor refocusing is going to take some time to implement. I have sent a few articles back already to our writers for minor enhancements, and overall the changes seem to be working. What you can do to help is spread the word about *TCJ* and send us your comments, stories, and questions.

I have been trying to find and contact vendors and service organization that support classic systems. A few still do exist, but will not last much longer on just local traffic. Since their numbers is getting ever smaller it is up to use to support them, by telling others of their services. If you know of some company supporting classics, please turn them onto *TCJ* and send me their name so I can add them to our mailing list. I have plans to provide several new reviews of support persons and organizations. To do this I need your input.

The contents of *The Computer Journal* will stay much the same as it is now. The number of hardware articles however has been considerably lower than I would like. Since new hardware projects are based on clone designs, it will take you the reader, who is using older systems to send us your articles. The current writers are heavily overworked on their own projects(some based on clones which is why their overworked) and so new writers and projects are needed. Drop me a note and let us talk about your projects. Not all projects need be full articles, our readers enjoy the short mini articles I run in Reader-to-Reader.

Well that pretty much sums up the directions of *TCJ* for the next ten years.

As always feel free to drop me a letter or message, I am glad to hear from you. Thanks for supporting *TCJ*!

Bill Kibler
The Computer Journal
P.O. Box 535
Lincoln, CA 95648
GEnie = B.Kibler.
CompuServe = 71563,2243.

NEW Regular Feature

Classic Support

Group Reviews

SUPPORT GROUPS FOR THE CLASSICS

By J. Wm. Weaver

If you are like me, it is quite a chore to locate groups supporting my interests and/or systems. For one, I live in a remote part of the California foothills, with few if any support groups located locally. So I have to travel to one of the two nearest cities, Stockton or Sacramento, approximately 60 miles to either one. Also, I have a large assortment of classic computers, from one of the first Altairs, to TeleVideos, to Commodore 64, to Osborne. With side assortments of Altos, Corvus, Sun to ATT 3B2 systems. With language interest from Assembly to BASIC to Forth, and operating systems from CP/M to DOS to OS/2 to UNIX.

Finding groups that support these, or any part of these interests, has been a tedious task. I finally settled on two of the many fine groups in the Sacramento area. More about them a little later.

For the many users, new and seasoned, who are looking for groups to contact, I will be creating a list, by regions, of groups and the systems supported, with names of people representing these organizations, phone numbers and/or addresses, to be contacted.

So if your group would like to be included, please send me a brief description of your group, systems supported, a little history, and if possible newsletters, with permission to print excerpts in *TCJ*.

Now to the two support groups in the Sacramento Area.

Sacramento Komputer User's Group

Meets the 3rd Thursday of each month, 7:30pm. Oakleigh Wedding Chapel 8452 Madison Ave, Fair Oaks, California. Contact Person: Richard Hughes Phone (916) 363-9198 Mailing Address: P.O.Box 214968, Sacramento, CA 95821

This is a fairly small group, with attendance fluctuating between 12 and 25 members, plus a few guest's. Informal but friendly meetings. History: This group began in the early days of micro's, to support the Kaypro computers (thus the "K" in computers). As time progressed, the interest changed. Currently, the group's interest span several CP/M systems, MS-DOS, OS/2 and UNIX. Has a library of Kaypro related software, and between its members, can usually find software for about a dozen different CP/M computers. Last year they got a write up in the local paper for repairing a Morrow computer after all the repair shops were unable to help the lady in distress.

Sacramento Forth Interest Group

Meets the 4th Wednesday of each month, 7:30pm.SMUD Training Building Room A, 1708 59th St., Sacramento, California. Contact Person: Bob Nash Phone (916) 487-2044

This is a very informal group, with attendance fluctuating between 4 and 18 members. History: This group was founded to provide support for the Forth Environment. Continues to support Forth, and has a wide range of experience in the industrial, commercial, and hobbyist applications. Always willing to help the neophyte. Most members also belong to FIG (Forth Interest Group) and each year the group journeys to the San Francisco Bay area for FIG's Forth Day seminars and the fireside chat with Chuck Moore.

Best wrap this up as I promised Bill K. that I would keep this first article to a minimum, and have it to him this evening.

To contact me (by snail mail) :

TCJ Group Support
c/o JW Weaver,
Drawer 180,
Volcano, CA 95689
Voice: (209) 295-3173
BBS: (916) 427-9038

I want to welcome JW to our stable of writers and supporters. JW is actually very skilled at fixing computers (I beleive he fixed the poor ladies computer last year) as well as writing, and with time I am sure that his column will become one of your favorite focal points when looking for classic support. BDK.

The Z-System Corner

By Jay Sage

Regular Feature

ZCPR Support

Looking Back

Looking Back: Z-System and *TCJ*

As I wrote in my last column, I had originally planned a special project for this issue: I was going to start from scratch with one of my computers and go through the process, as if I were a beginner, of installing NZCOM or Z3PLUS on it. As the first step of this project I was to update all the utilities on the distribution diskettes. From what I learned going through the installation, I expected to update the documentation, to help people over rough spots I discovered.

Well, owing to severe pressures at work, there just has not been time to complete this project. I did manage to update the utilities, and I did fire up a Kaypro that I extracted from the collection of old machines stored in my basement, but I have not had time to go through the installation process.

While trying to get an idea for what else I might write about for this special anniversary issue, I took out my whole collection of *TCJ*s and started to look through them, rereading some of my own and others' articles. This was a powerful nostalgic experience and brought back a lot of memories. Most of those memories were very sweet, because they reminded me of all the fun we've had and all the interesting things we've done in the Z community. I was amazed at how good and enlightening so many of the articles were. But some of the memories were sad, because they reminded me of people who are no longer active participants in the Z community.

It seemed quit appropriate at a major anniversary like this one to look back and take stock, so that's what I decided to do. In issue #54 I reviewed the ten-year history of ZCPR; this time I will trace the history of Z-System's relationship with *TCJ*.

The Earliest Days

My *TCJ* collection goes back to issue #19, dated July/August, 1985, almost eight years ago. That must be when my first subscription started, since it corresponds to an entry in my check register showing a $12 check written to "*The Computer Journal*" on July 9, 1985.

I no longer remember how I first found out about *TCJ*, but something I do clearly remember is being enormously surprised that a magazine so along the lines of my interests had been around for nearly four years without my ever having heard of it. Unfortunately, that situation probably persists to this day: there must be hundreds of people out there who would love to be reading *TCJ* if only they knew of its existence.

When did a connection first develop between *TCJ* and Z-System? Well, in looking through the issues just now, I noticed that advertisements started to appear in issue #20 for Echelon, a company created by Frank Gaudé to market the Z-System. Frank published a newsletter called "The Z Letter" for customers of Echelon, and most likely it was a brief reference there to *TCJ* that got me and a number of others started as *TCJ* subscribers. What a life-giving spark that turned out to be!

It was not until issue #23 that anything about Z-System actually appeared in a *TCJ* article (though I did see a reference to an article way back in issue #9 that apparently compared RPM and ZCPR). In issue #23 founding editor Art Carlson started what was apparently to be a new regular feature called "The Z Column". The second installment appeared in issue #24.

Very early on I proposed to Art Carlson that I write a regular column on Z-System topics. Art accepted my offer, and my first column appeared in issue #25, taking the place of his column. In those days there was some friction between the user community and Echelon over Echelon's understandable desire to exert some control over the Z-System utilities -- even the public-domain ones -- that it included in its commercial distribution packages. To avoid confusion and keep responsibilities clear, Bruce Morgen, Richard Jacobson, and I formed an organization called ZSIG (ZCPR Systems Interest Group) to promote new public-domain Z-System software development. My *TCJ* column appeared under the auspices of ZSIG and was called simply "ZSIG".

The first column mainly announced the formation of the new organization. Real technical content began with issue #26, where I talked about how to improve the performance of Z-System on a computer with only floppy disk drives. The ideas there are still useful today and, in fact, apply even to systems with hard disk drives. And to software other than Z-System.

With issue #27 there was a slight change in the title of my column (why, I can't remember) to "ZSIG Corner". The column dealt with command-line generators -- aliases and shells -- two of Z-System's most powerful features and the most important application of its mul-

tiple-command-line capability. The column in issue #28 covered an idea that originated with a *TCJ* reader, Dreas Nielsen, for implementing recursive aliases. His idea was much better than one of my own that I had described in an earlier column. Rereading this brought back the first sad memory: Dreas, like so many others, eventually moved on to other things, and we have not heard from him in many years.

Explosive New Developments

With issue #29 there was a major change. In my column in that issue I announced the surprising news that Richard Conn, the creator of ZCPR, had dropped out of the Echelon development team and that I had taken over responsibility for command-processor development. Moreover, the column announced the official release of ZCPR33, on which I had been working feverishly for several months but about which I could previously speak only in guarded terms, since nothing was official at that point.

In recognition of this change in my status, the title of my column was changed to "ZCPR3 Corner". Art Carlson actually got so carried away that he changed the affiliation listed for me from "ZSIG Librarian" to "Echelon, Inc." He apparently misunderstood what it meant to be a "member of the Echelon Z-Team". It did not mean that I had become an employee of Echelon. Both Echelon and my real employer might have been a little shocked at what they saw.

This had been an amazing period in my life. After getting home from work, I typically programmed until about four o'clock in the morning. Then I got up and got ready for work again at 6:30! At first, I expected that I might be able to do this for a few days, after which I would collapse. To my amazement, I continued on that schedule for several months. And felt terrific! It seemed that I really didn't need more than two or three hours of sleep each day. I had once met someone who claimed to sleep only one hour a night; now I could believe that this was possible. I still get by on rather little sleep -- but it's a lot more than two hours a night!

Developments proceeded at a very rapid pace during this period. The column in issue #30 mentioned a new product in development: NZCOM. Two issues later came the official announcement and full description of this amazing implementation of the Z-System. The actual release of NZCOM ended up being delayed while some important additional features were implemented, and it was not until issue #34 that NZCOM could actually be purchased.

My column in issue #31 provided the first complete documentation of my alias processing program ARUNZ. I still get a chuckle out of how I introduced the subject (J'accuse!), but you'll have to look at it yourself to find out why. Writing that documentation led to a slew of new ideas (teachers often learn more than their students do from the lesson), and two issues later I had to provide a further set of documentation. ARUNZ has been one of Z-System's most popular and widely used programs, but it probably also holds the world's record for the length of a beta-test period. I still have not officially released it!

An important milestone in *TCJ*'s coverage of Z-System came in issue #32, when Bridger Mitchell's column entitled "Advanced CP/M" first appeared. This became my favorite column and one from which I always learned a great deal.

In its second installment, in issue #33, Bridger announced and described another blockbuster advance in the Z-System: Z3PLUS, the Z-System for CP/M-Plus computers. Until then, the large fraction of the CP/M community that had computers (such as the Commodore 128 and Amstrads) running CP/M-Plus had been excluded from the Z community.

My column in issue #33 also covered some particularly interesting material. WordStar Release 4 had just come out. It was the first (and is still the only) program from a major software house that was written to take advantage of Z-System. Unfortunately, there were problems with the implementation. WS4 acted as a shell under Z-System, and this turned out to cause a good bit of grief. My column described a number of mistakes in WS4 and ways to fix some of them, and it started an ongoing discussion of ZCPR2- versus ZCPR3-style shells.

My comments about shells provoked the greatest controversy and volume of letters of any of my columns in *TCJ*. This led me to take up the subject again -- in greater detail and with greater force -- in issue #35. That column included some extensive patches to WS4 for improving its performance under Z-System. Deciphering code in overlay files is especially challenging, so this material might be of interest to people working on programs other than WordStar.

The Demise of Echelon

During this period, the Z community suffered a major loss. Frank Gaudé, exhausted from his efforts to turn Z-System into a viable commercial venture, decided to retire and fold Echelon. I announced this in my column in issue #36, but it must have been in the works for quite a while. I had not noticed it at the time, but in checking through my *TCJ* collection now I see that Echelon's regular "Z Sets You Free" advertisements stopped after issue #32. This meant that just as NZCOM and Z3PLUS were ready for market, Echelon was no longer there to market them.

Sadly, Frank Gaudé ended his effort in a state of great disappointment and discouragement, even bitterness. He retired to Lake Tahoe, and I don't believe anyone has heard from him since. He has never, to my knowledge, signed onto a Z-Node BBS since his retirement. When he quit, he quit totally. Maybe that's the only way one can do it after such intense devotion.

New *TCJ* Authors

Remarkably, during this same period there was a blossoming in the number of authors contributing Z-System-related articles to *TCJ*. In issue #35 Bruce Morgen got things started with a series

on modern assembly-language programming based on relocatable libraries.

In issue #36 Rick Charnes joined the roster. Rick just loved shells! In his first column he talked about named shell variables, a very powerful but rarely used feature of Z-System. Starting with the following issue, he actually named his column "Shells", though that wasn't the only subject he covered. Rick continued to write through issue #39. Though not originally a programmer, he had an intense passion for Z-System. This eventually led him, if I remember correctly, to change careers and become a computer professional. Since his work did not, of course, involve 8-bit computing, he eventually drifted away, and we have not heard from him in several years. I miss him.

Hal Bower contributed to four consecutive issues starting with #37. In the first two, jointly with Cam Cotrill he described ZSDOS, their new BDOS replacement. Hal and Cam always do things with utter thoroughness, and ZSDOS is still the state of the art in disk operating systems for 8-bit computers [but ZSDOS2, a banked version of ZSDOS, will soon advance the art further]. In his next two articles, Hal continued the theme started by Bruce Morgen on advanced assembly-language techniques with a discussion of PRL (page-relocatable) files.

Cameo Appearances

Over the years there have been cameo appearances in *TCJ* by a number of well-known personalities in the Z world. Starting with issue #42, we had one in nearly every issue. Dreas Nielsen, whom I mentioned earlier, contributed an article on dynamic memory allocation for that issue. For the next, Michael Broschat, an ardent Z-System user, wrote an article on the S-100 bus. Like Rick Charnes, Michael ended up becoming a computer professional (his area had been Chinese!).

There was an especially exciting cameo appearance in issue #43. Rob Friefeld, who has authored some of the most valued programs in the Z-System arsenal, described LSH, his exceptionally fine history shell. I use many kinds of computers in my work -- micros, minis, and mainframes -- and I have never seen another history shell that holds a candle to LSH. Three of Rob's other programs -- SALIAS, SCOPY, and XOX -- are also among the highest quality and most popular Z-System tools.

For issue #44 Dan Mareck wrote a review of Bridger Mitchell's remarkable DosDisk program, which allows CP/M computers to work with MS-DOS DSDD disks, even those with files in subdirectories.

Issue #45 offered some really special excitement. Many authors had described Joe Wright's brilliant work; in this issue Joe appeared in person. His company, Alpha Systems, had acquired from Borland the right to market the 8-bit version of Turbo Pascal. Naturally, Joe was eager to make it Z-System compatible. Joe's agreement with Borland did not allow him to make any changes to the code, but in his usual clever way, Joe figured out how some simple patches and a toolbox of Pascal subroutines could accomplish the goal.

It looked as though we skipped over issue #46, but David Clarke, in his column on Modula 2, was just setting things up for his follow-on in issue #47 that told how to Z-ify Modula 2. Almost all high-level languages were covered now. Back in issue #38 I had announced an update to BDS C that included full Z-System support. Issue #48 carried a review of it by Carson Wilson, a major figure in the Z world. Carson wrote the Z80DOS disk operating system replacement before he teamed up with Hal Bower and Cam Cotrill to develop ZSDOS. He also wrote the very popular ZDE (Z-System Display Editor).

Stress Takes Its Toll

Even during that golden age of Z-System, we suffered some major losses. Two were particularly painful for me, because these two people were a source of great enjoyment in my Z-System work.

One was Richard Jacobson, whom I mentioned earlier and who had had a long, behind-the-scenes connection with *TCJ*. Way back in issue #30 Art Carlson announced that Richard's Lilliput Z-Node had become the official BBS for *TCJ*. Although Lilliput was a pay system, *TCJ* subscribers were offered free access. To inaugurate the new service, Richard wrote an article for that issue describing Lilliput.

Richard had a wonderful sense of humor, and many Z-System aficionados loved the discussions on Lilliput. He attracted a strong following in the local Chicago community, much the way Lee Bradley has in Hartford in more recent times. In issue #38 I had the pleasure of announcing that the Lilliput Z-Node had become the new Z-Node Central, the core node for the support of Z-System.

Unfortunately, soon thereafter Lilliput was beset by a rash of major hardware failures. In issue #42 I had the sad task of announcing that Lilliput would not be rebuilt. Al Hawley's Ladera Z-Node in Los Angeles became the new Z-Node Central. With his own Z-System focus lost, Richard's presence, too, was soon absent.

The other deep loss for me personally was that of Bridger Mitchell. Issue #45 was the last one to carry the regular column he had started back in #32. That streak of 14 columns has been surpassed, I believe, by only two other columnists (I don't count editor Art Carlson here). I am one; this column is my 36th. Our current editor, Bill Kibler, is the other. I don't know when he started writing for *TCJ*, but I have never seen an issue without his "Computer Corner".

I missed Bridger's contributions so badly that I tried very hard to drag him back. There was a hopeful sign when an "Advanced CP/M" column appeared in issue #54, but there has been no follow-on.

Stress finally took its toll on publisher/editor Art Carlson at about this same time. Art had published 47 issues of

TCJ when Chris McEwen took over with issue #48. This represented a more intimate connection between *TCJ* and Z-System, since Chris was a Z-System person himself. He was the sysop of the Socrates Z-Node in South Plainfield, NJ, and had contributed an important article in *TCJ* issue #42 entitled "Using BYE with NZCOM". I'm sure it helped a number of people set up new remote access computer systems (BBSs).

New Contributors

In that same issue #48 when Chris took over, two new regular columnists joined *TCJ*. My friend Clif Kinne, at my urging, started a column on macro programming for the ZMATE/PMATE text editor. These columns appeared through issue #52. At that point, having gotten little reader feedback, he did not know where to go next.

The second new feature was Bill Tishey's "Z-Best Software". This became one of the most popular columns in *TCJ*. Way back in issue #36 I had spotlighted Bill as a shining example of how a non-programmer could contribute in an extremely valuable way to the development of the Z-System community. Bill had taken on the duty of librarian for ZSUS, the Z-System Software Update Service. This service made it possible for people without ready access by modem to Z-Nodes to get the latest software releases on diskette.

Bill scoured the Z-Nodes across the country for new programs and new releases of old programs. But he didn't just collect the files; he also built up an extensive system of documentation. In his columns he shared his knowledge of Z-System software with *TCJ* readers. The columns continued through issue #53. I'm not sure what happened then; I think Bill just got too busy at his job. I tried to contact him several times by phone and letter, but I never got an answer.

What Does the Future Hold?

I can't really answer that question, of course. Z-System contributions to *TCJ* have gone through lots of ups and downs, but the overall level of vitality seems never to have seriously wavered. Some authors have departed, but others have come on the scene. We don't have the same volume of new software developments that we once did, but we've had real surprises on the hardware front. Who would have imagined that a burst of articles would appear on new 8-bit computers, but we've seen two new machines -- the YASBEC and the CPU280 -- presented in the pages of *TCJ* in the past year and a half.

We seem to be in an especially difficult period now. With the poor economic climate, everyone -- all over the world -- has had to spend more and more time trying to keep afloat at his/her real job. This has cut deeply into the time available for hobby activities, which is certainly what 8-bit computing is for all of us.

I feel the same pressures myself. It has become a real struggle to find the time for this column. I think it is partly fear that keeps me going. In my chronicles above, there haven't been any cases of people who just pulled back a little; once they let go, they seemed to disappear totally. I'm not ready for that. There is still a lot to talk about, and I hope to be around well into the second decade of *The Computer Journal*.

SUBSCRIPTION RATES for *The Computer Journal*

These rates are effective ***January 1, 1993.*** Foreign rates now reflect the actual charge of additional mailing fees and have been set by adding those charges to current subscription fees. **Sales tax is no longer needed for California residents on mail order subscriptions,** taxes are collected however on items purchased by mail, such as back issues and floppy disk programs.

Subscription	U.S.A.	Canada/Mexico	Europe/Other Countries
1 Year Surface	$24.00	$32.00	$34.00
1 year Air	$34.00	$34.00	$44.00
2 year Surface	$44.00	$60.00	$64.00
2 year Air	$64.00	$64.00	$84.00

All U.S.A. shipping is third class bulk mail, except First class or air which cost an additional $10.00. Foreign surface and Foreign Air Mail is currently shipped as printed matter and packaged in appropriate mailing envelope. Current rates are calculated on average shipping weight of 6 ounces.

Dr. S-100

By Herb R. Johnson

Regular Feature

Intermediate

Letters and Future

My apologies to my readers for a short column, but my wife and I are house-hunting. I can tell you, it's as hard to find a nice house as it can be to find a good S-100 system. You can search long and get a bargain, especially if you're willing to fix it up; or you can grab whatever comes across your path and hope it works; or you can pay top dollar -- and hope it works!

Letters & Correspondence

I got several notes of encouragement and praise for my recent column on diagnosis and repair of my S-100 system on the FidoNet **CPMTech** echo (BBS message exchange), most notably from **Fred Hatfield**, a friend of long standing and (I believe) the Echo moderator for the **Shortwave** echo.

I thought I'd share the stories I've received about other people's experiences in repair, construction, and purchase. For clarity and further wisdom, I have added editorial comments in square brackets.

David Drew of Newark, Delaware wrote:

"It was a delight to read your recent feature in TCJ. I thought you might be interested to know who some of your audience are these days. My involvement in S-100 systems has been a hobby for learning how computer software and hardware works.

"My first S-100 system was acquired in 1979. I built the SD systems SBC-100 CPU card, Versafloppy I [floppy disk] controller and ExpandoRAM memory kits, starting with 32K RAM. A 2.5 MHZ Z80 [5 MHz clock] seemed plenty fast to me. The disk drive was an 8" SSSD Shugart 801 sitting on a shelf - no cabinet. The power supply was a kit by Sunny. Cabinets were too expensive for me, so a scrounged one-half of a bare 22-slot motherboard that someone had neatly cut in two [into an 11-slot motherboard!]. I mounted it in an open cardboard box with a small fan blowing over it.

"I found a dead Hazeltine 2000 terminal that was to be scrapped, and repaired it. The repair cost $50 for the service manual, five evenings of my life in front of an oscilloscope, and 79 cents for a replacement IC from Radio Shack. I used that system for about four years before I bought a second Shugart drive and built a wooden cabinet for the two of them. I still kept the cardboard box, however. I spent those years learning 8080 assembly language, writing some Ham [amateur] radio programs to send and receive Morse code and radio teletype (my ham call is K3DX), and creating some accounting programs for my church. Later I installed ZCPR2, after disassembling my BIOS. I reluctantly sold the Hazeltine 2000 at a flea market last year [1991]: it still worked.

"In 1987 I bought my second S-100 system for $50 at another flea market. Not only did this one have a cabinet, but it had the 4MHz [Z80 SD Systems] SBC-200 CPU card and the double density Versafloppy II controller card. I was in heaven! The $50 deal also included a Hazeltine 1500 terminal, a SummaGraphics [digitizing and mouse] bit pad, a MicroAngelo graphics card (very expensive in its day!), a monochrome video monitor and a separate keyboard. It was sold "as is" because it didn't work. But it did work just fine once I installed a fuse in the empty fuse holder.

I think it was about this time I upgraded to ZCPR3, disassembled the BIOS, and added SSDD capability. I also wrote a Turbo Pascal program to receive weather FAX maps by shortwave radio and which would display the maps using the MicroAngelo graphics card.

In 1990, a friend who had upgraded to a PC-AT clone gave me his S-100 system, which had 3 Mitsubishi DSDD 8" [half height] drives and a 256K "Light Speed 100" RAM disk made by Digital Research Computers [not to be confused with Digital Research Inc., who distributed CP/M!]. I now had 1.2 megs per floppy -- what storage capacity! This same friend also gave me his entire library of 70 DSDD 8" diskettes and 90 5.25" NorthStar diskettes. Last year [1991], I acquired a NorthStar system with a hard disk for $10 at a flea market. That monstrous 18 Mbyte [8"] Winchester is a big, heavy, noisy power hog. I wonder if the old NorthStar controller could work with any of today's modern hard drives? [Probably not -- the data lines are not compatible -- HRJ.] I doubt it, but so far I haven't had the time to research this.

Over the years I've also collected an assortment of S-100 cards, most of them working, some not. I'm sure you and some of your readers remember them: the ByteSaver [2708] PROM blasters by Cromemco; a QT clock/calendar board, so poorly done that it is surprising that it actually sold for $100 at one time; a

1200 baud modem card, Mullen [S-100 bus] extension cards; and an 80-character [by 24 lines? 12 lines?] video board by Solid State Music (no manual, sigh) [I have a manual!]. Miscellaneous cards include serial, parallel, and 2708 EPROM cards. My two prizes are a high-speed, 16-channel, 12-bit A/D converter card by Techmar; and a Cromemco TU-ART card which, in addition to having two very flexible serial ports, has numerous programmable and interrupt-driving interval timers. A few of these boards, like the TU-ART, don't work, which explains why they were given away! I have the manual and hope to fix the TU-ART card.

I'm intrigued by your idea to make a hard disk controller board. I'd like to toss out another idea: Is it possible to use existing S-100 parallel I/O boards (with appropriate software) as SCSI interfaces[or IDE - BDK]? How many lines does the SCSI interface require? Is it TTL compatible? What would we do for BIOS software? Where is the best place to find documentation? Would a mere 4MHz Z80 be fast enough to keep up with the drive?

Sorry I didn't send this to you by Email, but I have access only to Internet. My net address is DrewD@engg.dnet.dupont.com. I've logged in to the Drexel Hill [Pennsylvania] NorthStar BBS from time to time, and just started to use the Philadelphia Area Computer Society (PACS) BBS. I'm in Princeton a few times a year to visit my inlaws, and of course there is the annual Trenton Computer Festival near you. Perhaps someday we can meet.

Regards, Dave Drew
PO Box 1050
Newark DE 19715

Thanks Dave for your detailed letter! Most of his experiences are representative of the history of many S-100 owners, recent and past. I'll address his questions about SCSI at a later time, but they are right on the mark!

The other letter of note is from a long term S-100 owner, John Haugh of Shorewood WI, who maintains a fleet of Cromemco systems. For those of you who think S-100 systems are "primitive", read on:

I was happy to see your advertisement in Nuts and Volts, since I had lost your address after we did business the last time, when I ordered about six or more [Cromemco] 64FDC [floppy disk controllers]. They were very satisfactory and all are now in use.

Sometimes I think I must be the last person on the planet still using Cromemco systems but they serve me well and I am still looking for sources of supply. I have eight children and as often as one of them goes back to college for a masters degree or PhD, I am able to supply a free computer for producing papers and research data. With sixteen grandchildren, I hope to be in the old computer supply business for a long time to come.

[John describes some cards he'd like to buy, and lists some he'd like to find. The latter list is instructive of the kinds of technology Cromemco has developed.]

CPU cards
XXU XMU XPU XMM 68010 CPU's & page demand controllers, UNIX
Memory cards
MCUX controllers for XXU systems
MCU controllers for DPU (Z80/68000) and XPU
MSU 1024, MSU 2048
MSU 4096, MSU 8192 error correcting memory needing controllers
8192KZ, 4096KZ,
2048KZ, 1024KZ
256KZ non-error correcting memory, no controllers
64KZ for Z80 CROMIX, CDOS only
Disk controllers
ESDC for EDSI disk, SCSI tape
STDC for ST506 (MFM) 5" drives
SMDC for SMD drives
WDI II for obselete IMI 5" drive
64FDC for 8" and 5" floppy, double density
16FDC for 8" and 5" floppy, Persci
4FDC for SSSD, convertable to DSSD
I/O cards
OCTART 8 serial ports
IOPX XXU I/O controller
IOP XPU and DPU I/O controller
CSP for 9-track tape drives
CNET network controller for IOP, IOPX
QUADART 4 serial for IOP
8PIO 8 parallel ports
4PIO 4 parallel ports
Analog I/O
ADC12, DAC12 A/D, D/A; 12-bits
D+7A A/D, D/A; 8-bit
GPIB HP-IB or IEEE-488 instrument bus
(Broadcast Studio) Video controllers
SVID Color video generator
SDMB DMA controller for 2-port memory
SDMA earlier version of above
1024KTP, 256KTP 2-port video RAM
SDCM Color modulator
SFLSH Flash digitizer
SDD older digitizer
SALPH multi-plane blender

The systems I have at present include one CS-400 tower with ZPU, XMM, MCU, 4MBytes of memory, 2 OCTARTS, one 50MByte hard drive, one 5" floppy, one 20MByte cartridge drive, two Cromemco 3102 terminals and a Diablo 630 [daisy wheel] printer with dual bin page feeders. This system runs both Cromix 167 [a UNIX work alike] and UNIX System V. Software includes C, Fortran, assembler, and a relational database, and WordStar 4.0 which I have adapted to run under Cromix. [John describes other systems of equal capability.]

I mention these systems for your information because I am willing to be a knowledge base type resource to any of your customers if it helps. I also have a complete set of the Cromemco user group publication "I/O News" which has many helpful pieces of information.

[John also has a few small floppy-based systems he could sell, configured for CDOS; and some non Cromemco S-100 cards and systems; and some S-100 boxes (bus and power supplies).]

Yours very truly
John J Haugh M.D.
4205 N Newhall St
Shorewood WI 53211

Like many current (and former) S-100 owners, John is very willing to help others with their systems. One of my pleasures in this "business" is the conversations I have with people like John and others. If you send a letter, consider adding a note permitting me to publish it in my column and you too may be a Doctor of S-100-ology!

Future columns

David Drew recently bought from me an SD Systems Versaflopy II floppy disk controller. I have many of these in stock. It is David's hope to make his work with his IMSAI 8080, which some of you may remember originally included a 1MHz (not 10!) 8080 Intel processor [was last months center fold BDK]. This will be a challenge, but at least the BIOS I rewrote for it is 8080-compatible! There seems to be a minor bus compatibility problem, not necessarily a speed problem, but we'll both test the possibilities. As floppy disk control is a design problem across all disk-based systems, I'll probably devote a future column to it.

It's hard to write about the future of S-100 systems beyond saying the obvious: there is none! They're all obsolete! etc. But this is either annoying, boring, or a horrible lie (depending on your tastes), so I thought it through a bit further and decided to write on....

The rise and fall of S-100 systems

In the beginning, owning a computer was a dream only of engineers and programmers who had the rare and profitable privilege of using them at school or work. A few "techies" actually had them in their basements, thanks to a surplus sale, or auction, or just giveaway of ancient accounting machines or old university instruments or industrial surplus. The first personal computer market was made of people like these.

The first Altairs were sold in 1975, with 1K of memory and a front panel of switches and lights for $400. Not long after came IMSAI and an "alphabet soup" of other companies with S-100 products. Along with single board computers like the Apple II, KIM, and so on, there now existed a common ground of several computers that people could trade software, hardware, and information. Computer clubs and garage companies grew rapidly and sold thousands of units to the hobbyists throughout the 1970's.

Where the hobbyists led, the entrepreneurs followed. Early business systems were developed, generally in BASIC, and enjoyed various degrees of success. The S-100 system was ideal for this market: massive boxes and easy expandability made such systems look "industrial". In fact, S-100 competed well at the low end with other industrial bus-based systems. Into the 1980's, industrial, business and software development systems contained faster Z80's, followed later by 8086, 68000 and 80286 processors running Turbodos, MS-DOS, Concurrent DOS, and various forms of UNIX.

While the commercial and industrial market continued into the mid-1980's, the personal S-100 market probably peaked in the early 1980's. Z-80 based systems continued to be popular with both hobbyists and small business, but it was in the form of "transportables" like the Osborne I and the Kaypro. Even Apple II's often had Z-80 cards, especially when Digital Research bundled them with CP/M 3.0. An enormous (at the time) quantity of software around the Z80 and CP/M came into being at this time. S-100 systems, even slimmed down to a few slots, were too "clunky" except for the Sol and other "integrated" systems designed for the desktop. It was about this time, incidentally, that the S-100 manufacturers got together with the IEEE and created the IEEE-696 bus standard, which established the computing tradition of standards that are accepted only after they are made obsolete.

It was clear by the mid 1980's that the IBM PC, and its 100% (not 60% or even 95%) compatible clones would be the machines of "the future", along with the totally CP/M incompatible Macintosh. Such was the momentum of the Z-80 (and Apple II) world, and the relatively poor performance of the IBM PC and Mac, that it took perhaps three years after their introductions to become leading systems (in terms of sales and price/performance).

So, the S-100 bus began to return to its traditions of industrial control and hobby development in the late 1980's. As systems became obsolete, and as "obsolete computing" itself became obsolete, S-100 systems got cheaper and more available on the surplus market. However, the IBM PC again competes with these systems as the old XT's and now AT's appear as surplus, thus setting a value "ceiling" with the phrase: "I can buy a used 'PC for that price!".

Today

S-100 systems are giveaway items, when you can find someone to take them. Without surplus shops, scrap dealers, or a techie constituency a system literally ends up in the dumpster more often than not, or "sold for the price of shipping, to give it a good home." Docs are unavailable from anybody (almost!). 8" disks and drives look ENORMOUS by today's standards. And yet... and yet my experience is that most of these giveaway, neglected systems still work! They are more-or-less decipherable, and actually perform reasonably well at word processing, control, and general knock-around tinkering and self-education. Software is still kicking around for programming, desktop publishing, communications and BBS systems, and more.

Tomorrow

The big question is: what will become a collector's computer? What will be valuable to own, and what will be junk? I have no clear response, but I do have several experiences from buying and selling S-100 systems:

1) IMSAI's have great sentimental value. To the people who learned programming and electronics on them in the 1970's, or whom built their businesses on IMSAI's, these systems are worth a few hundred dollars, maybe more.

2) Altair's were the "first" computer. To people under the age of 35, these are ancient history. They want one "in per-

fect condition" to blink lights on their mantelpiece. But, only as a curiosity, so they won't pay very much.

3) REAL antique collectors have zero interest in any computers. "No market" they say, i.e. it's not listed in Collector's Weekly or whatever journals they follow.

4) Apple I's are worth their weight in gold. However, since no one can ever find one, this hypothesis has not been tested. (I know, an Apple I is not an S-100 computer. But I keep getting asked, anyway.)

5) Many people who learned computers before 1980 would rather not learn another one, and so they will buy Kaypros (or Osborne's or NorthStar's) forever. This hypothesis may also apply to IBM or Mac owners, but until those systems become totally obsolete we'll never know. Anyway, these people will also buy and play with S-100 systems for the same reason.

6) There aren't a lot of simple and cheap NEW computers around. There are plenty of single-board controllers, with no disk drives); and IBM clones that are too complicated and have no technical docs anyway. Old CP/M transportables are ok, but the hardware is rather inaccessible for SERIOUS tinkering. S-100 systems are cheap (free almost), well-documented (when you have docs), and are easy to figure out (relative to ASIC-packed PC's). You can even REPAIR them! and learn from the experience!

Alternative futures

That's the future, as I see it. There are other futures, of course. Clearly the current generation of programmable logic devices will be a tinker's heaven; when the tools are available for surface-mounting chips in your basement. When the programmers for these devices cost less than a '386-25 system. And, when the software for programming them doesn't require a '486 system to run and can be bought at Radio Shack for $99. All this presumes, of course, that enough people know which end of a soldering iron to hold, and that they have some notion to program hardware instead of "building" C++ object libraries.

OR

When you can buy $10 or even $5 "Chicklet computers," each of which will run a device, and you can string them together like Christmas lights, and program them all from your favorite computer.

OR

We all become multimedia mavens and send five-minute epics to "America's Funniest Home Media" for a chance to own the Altair Virtual Reality 8800, with 3-D simulated toggle switches and laser lights that blink at 100 MHz.

OR

(Choose your own version of near-future computing technology, and send it to me. I'll send S-100 cards to the best, and publish the rest! Irony is required, and it should have some hint of reality to it. Don't forget your bio.)

Herb's address is CN 5256 #105, Princeton NJ 08543. Or call (609) 588-5316 and ask for "Dr. S-100."

CLASSIFIED, FOR SALE and WANTED

Amstrad (c) PCW SIG: $9 for 6 bi-monthly newsletters dealing with the most popular CP/M machines still in production. Learn where to buy 3" discs, how to add 3.5 and 5.25 drives and where to buy the 8 MHz Sprinter board with room for 4 Meg of RAM. Make checks out to Al Warsh, 2751 Reche Cyn Rd #93, Colton, CA 92324.

For Sale: GIMIX 6809 SS-50 floppy disk controllers. Have six to sell at $25 each plus shipping ($5). Bill at TCJ (800) 424-8825.

WANTED: BOOT disk for Intertec Data systems, Compustar model 40 Video processing unit. Write to Roger Olson, 2304 West 4th, North Platte, Nebraska 68101.

WANTED: Chess Programs. I am trying to collect OLD chess programs such as Microchess 1, Sargon 1, Mac Hack 6, etc. Any computer, any format, any language. Prefer with source. Mr. Carey Boodworth.

WANTED: OLD K&R C complier with 'C' source code. Nothing fancy, plain K&R, not ANSI C and NOT Small-C. Carey Bloodworth.

WANTED: Forth written in portable 'C'. I've heard that one exists, but I've never found it. Prefered disk formats for all three wanteds is: MSDOS 360K 5 1/4, TRS-80 COCO RS-DOS or OS9. I can not pay much more than postage, sorry. Mr. Carey Boodworth, 1601 North Hills Blvd., Van Buren, AR 72956.

WANTED: Person or group willing to do small runs of PC boards for *TCJ* projects. Double sided boards cost too much from regular board houses when runs are 10 to 20 a year. Current needs are for small Kaypro adapter boards. Will pay for services, but amount is very low. Contact Bill at *TCJ*, or Chuck Stafford at (916) 483-0312 (eves).

The Computer Journal classified section is for items FOR SALE. The price is based on Nuts & Volts rates. If you currently have a Nuts & Volts ad just send us a copy of the invoice and we will print the ad for the same price.

Classified ads are on a pre-paid basis only. The rate is $.30 per word for subscribers, and $.60 per word for others. There is a minimum $4.50 charge per insertion.

Support wanted is a free service to our readers who need to find old or missing documentation or software. Please limit your requests to one type of system. Call *TCJ* at (800) 424-8825 or drop a card to *TCJ*, P.O. Box 535, Lincoln, CA 95648.

Four for Forth

Special Feature

Rochester Conference

Forth Microprocessors

By Jack J. Woehr

The 1992 Rochester Forth Conference, held at the University of Rochester, New York, was the twelfth of its kind. Since 1980, experts in the application of Forth to real world problems have been meeting and presenting timely papers on recent developments. This year's assembly, whose theme was Biomedical Applications, saw the announcement of four new Forth-oriented microprocessors. Two come from the United States, one comes from Germany, and one from Russia.

History of Forth Microprocessors

Forth is a language originally designed for embedded systems programming which has been ported to nearly every computer platform, from tiny microcontrollers to the IBM 370 and to Cray supercomputers. Interesting Forth-coded applications include:

- the hand-held gizmo carried by every Federal Express delivery person;
- the air traffic control system at the Riyadh airport in Saudi Arabia;
- IBM CAD
- VP Planner
- any number of space projects by NASA including the NASA MPP (Massively Parallel Processor);
- commerical satellite projects by Orbital Sciences Corporation.

Forth is modelled on a dual-stack architecture. Instead of possessing execution frames like C and Pascal that mingle arguments with return addresses, Forth pushes subroutine arguments to the data stack and return addresses to the return stack.

The Forth virtual machine posits subroutines consisting of address lists of other subroutines, and so on, until the lowest level at which machine-coded primitives are reached and executed. Excercises in Forth execution enhancement take two distinct tacks: optimizing the compiler for use on CISC or RISC processors, and designing special microprocessors to execute Forth more efficiently. (This dual and opposite approach to virtual machine efficiency has also been seen in LISP.)

Early in the 1980's, Rockwell issued the 65F11 with the Forth kernel in mask ROM. This processor had no special Forth-oriented instructions, but was very popular as a control development platform and stimulated interest in microprocessors designed specifically for Forth.

In 1985, Zilog began producing the Super8 which had microcode instructions supporting the Forth virtual machine (ENTER EXIT NEXT). There was no special support, however, for dual stacks: the overall architecture remained Z8-like CISC.

The first true dual-stack Forth microprocessor was the Novix NC4000 (1986), designed by a team headed by Forth inventor Charles Moore. The chip was produced in sample quantities, and the design eventually sold to Harris Semiconductor, where it became the core of the RTX2000 family (1988). The RTX2000 was fast to the tune of about ten (10) FIPS (10 million Forth Instructions Per Second) at a 10 MHz clock. Harris marketed the RTX2000 agressively for a few years as the answer to execution-intensive integer control algorithms, but during an organizational retrenchment two years ago discontinued the marketing, though not the production, of the RTX2000.

1986 also saw the design of the WISC, Inc. CPU-16, a dual-stack writeable instruction set CPU designed in discreet logic.

From 1979 onward, a team at Johns Hopkins Applied Physics Lab were engaged in design efforts centered on efficient Forth execution in custom microprocessors for the various projects connected with the space program. Starting with the AMD2900 bit-slice engine, and progressing to custom silicon, their effort culminated in 1988 in the FRISC-3 (Forth Reduced Instruction Set Computer), a dual-stack RISC chip marketed as the SC32 by Silicon Composers of Palo Alto, California and henceforth referred to by the latter name.

Other designs have been proposed and have reached varying stages of fruition (MISC M17, Chuck Moore's SHBOOM and uP20, etc.), but the above are the most notable of the first generation of Forth-oriented microprocessor designs. Now we will proceed to the new announcements made at the Rochester Forth Conference.

FRP 1600

Klaus Schliesiek-Kern is a tall, thin, graying, fortyish imp who wore a headband with feathers and a bemused smile throughout the entire conference. He was a significant and unmistakeable voice in the late-night political debates in the dormitory lounge between the Americans, Canadians, English, Germans, Russians and others, fascinating conversations which sometimes lasted until after dawn.

Herr Schleisiek-Kern is a principal of DELTA t GmbH in Hamburg, Germany,

who attended the Rochester Forth Conference to present a paper on the FRP 1600 Forth RISC Processor. The FRP 1600 is being marketed as a microprocessor optimized for high performance in realtime control applications.

The FRP 1600 was originally designed for inhouse use at E-T-A GmbH, a German company whose primary concern is the manufacture of circuit breakers. Optical quality monitoring on production machinery designed inhouse had become a critical concern by the time the design team met in 1988. Members of the team included Mr. Schliesiek-Kern, Gerhard Erbacher and Norbert Schumann.

The initial design criteria for the FRP 1600 were the following:

16-bit processor with a Forth architecture.

Memory bus and interrupt structure compatible with the Motorola 68K family.

Dual stacks residing in external memory for fast context switching (compared with the totally on-chip stack of the RTX2000, which must be copied out to external RAM during a task switch).

Minimal pin count.

Hardware support for image processing applications.

The design criteria were met by a 1-micron CMOS sea-of-gates design running at 15MHz. Every instruction executes in one clock cycle, with the exception of fetch, store and long literals, which take two clocks. The address of the next instruction is placed on the bus while arithmetic operations take place.

The FRP 1600 has a 20-bit address bus and a 16-bit data bus. All addresses are word addresses which allow 16 64Kword pages of memory. Programs may only reside on the zero page, but stacks may reside anywhere in memory.

The FRP 1600 uses a familiar instruction decoding trick used by several Forth microprocessors and introduced by the Novix of using the most significant bit of the word-length instruction to differentiate between subroutine references and other instructions. When bit 15 is zero (0), the instruction is actually a subroutine address shifted right one position (meaning that all subroutines must commence on an even address boundary: the address is left shifted one place by the instruction decoder). If bit 15 is a one (1), an ALU or other machine instruction follows in the lower 15 bits. This allows a flattening of the classic Forth virtual machine from lists of addresses pointing lists of machine instructions into lists of interleaved addresses and machine instructions.

Interrupt logic includes RESET, NMI, INT (maskable) and IACK (Interrupt Acknowledge) signals. Interrupts are serviced after at most three (3) cycles.

A single Eurocard prototyping board is available with an optimizing Forth-83 system featuring a pre-emptive, prioritizing multitasker. A development environment by MicroProcessor Engineering, Ltd., of Hampstead, England is also available. The producers are committed to ANS Forth when that proposed standard is approved.

FRP 1600 Contact Information:
DELTA t GmbH
Uhlenhorster Weg 3
D 2000 Hamburg 76
Germany
email: delta@cix.compulink.co.uk

FOREX 1620/1624

Pre Boris Katsev and fils Sergei Katsev held the record at the Rochester Forth Conference for distance travelled to attend, arriving from St. Petersburg in the Russian Republic. There they are the proprietors of FOREX, a private enterprise dedicated to exporting for sale the current product of one of the few domains of technology wherein the former Soviet Union kept fairly close upon the heels of the West, that of chips, board and software. FOREX is particularly interested in Forth, which provides rapid prototyping in a difficult economic and work environment wherein if anything good is to happen, it had better happen right quickly.

Sergei speaks very good English; his father, Boris, speaks almost none. Both are cultured, well educated, and speak Russian with a precise and mellifluous accent to make the average high school Russian teacher weep with envy. I asked Sergei, who is about my age, what his college years in the Brezhnev era had been like. "For me and for my friends in the university, the 1970's were like the sort of dream where you struggle to wake up, but keep falling back asleep, helplessly," he told me.

The FOREX 1620/1624 is a pair of dual-stack Forth-oriented microprocessor designs in search of a market, possibly a patron. Somewhat resembling the Novix/RTX in overall architecture, the 1620 addresses 1Mbyte of memory and the 1624, 8 Mbytes, with byte as well as word addressing. Each microprocessor has eight (8) general-purpose registers and seven (7) system registers (stack pointers, segment registers).

The instruction set contains four classes of command: CALL/RETURN, 8 branching, instructions, Read/Write instructions for 8/16 bits, and ALU instructions (16 arithmetic-logical, 8 shifts, 8 compares). Most instructions are 1 or 2 cycles; multiplication is a 16-cycle instruction. There is one external INT line and one internal, non-maskable interrupt from the on-chip timer.

The FOREX 1620 comes in a 48-pin DIP package and the 1624 in a 52-pin PLCC. Both CMOS, they are being marketed for embedded control applications. FOREX suggests that reserved instructions may be used to extend the instruction architecture for custom designs based on the 1620/1624.

In addition to presenting at the Rochester Conference a prototyping board based on an already well-known Russian clone of the original Novix, complete with MS-DOS based cross-development environment, the Katsevs showed us chip-on-board printed wire boards that were extremely attractive and well-done. Examining closely one of the boards, I asked

Sergei, "This is all hand work, isn't it?"

"Yes," he replied. "It is a process suitable for automation, we think."

There seems to be a gap between layers of technology in Russia. FOREX engineers lay out printed wire boards in advanced CAD software, but the board house doesn't have tools which can read the file formats output by CAD packages. Instead, designs are plotted and films re-created from the plots by technicians at the board house using older, more labor-intensive methods.

Most difficult to explain for the Katsevs were the nature and the scope of their marketing plans. They are at present in the exploratory stage, and could benefit from guidance from potential customers abroad. In one particularly confusing exchange in the cafeteria line, I was unable to extract the information I wanted from a willing but puzzled Sergei. Sensing an obstacle rooted somewhere other than in the small language barrier, I turned for assistance to Boris Nashanchik, president of Amics Enterprises, Inc., Rochester, New York, who was in attendance at the Rochester Forth Conference.

"What I'm trying to ask him," I explained to the patient Mr. Nashanchik, "is how they compute their cost-of-goods-sold."

Mr. Nashanchik pursed his lips in a wry smile. "They don't," he replied. "They have no idea."

"I can imagine what they teach doctors in medical school in Russia. But what do they teach people in accouting school over there?" I asked.

"How to count! How to count other people's money," he answered, laughing.

In addition to Forth microprocessors, FOREX also has available a number of software packages seeking marketing outlets and distributors in the West, including a very high level MS-DOS hosted 8051 software emulator.

FOREX 1620/1624 Contact Information:

FOREX
59, Bolshoi Pr., P. S.
St-Petersburg, 197101
Russia
email: root@forthi.spb.su

WISC CPU-16/CPU-32

You can probably have more fun with the Writeable Instruction Set Computer CPU-32 than with any other processor mentioned in this article. Afterwards. you may wonder whither you have strayed, but the trip is probably worth it.

I explored the WISC CPU-16 in a Dr. Dobb's Journal article (see Bibliography at end of article), a 16-bit discreet-logic microcoded central processing unit designed as a drop-in card for the PC/AT bus offering a dual stack architecture and a completely writeable microcode cache. WISC Technologies (now a division of Epsilon Lyra) used about half the microcode pages to implement Forth, which left half the pages for play. I used the empty pages to implement a model PROLOG engine.

Dr. Glen B. Haydon, a retired medical doctor and well-known Forth theoretician and pedagogue, is president of Epsilon Lyra. He describes his foray into microprocessor design as "less expensive for a retiree than a country-club membership", and had two bits of news to dispense at the Rochester Forth Conference Firstly, with the assistance of Rick van Norman (formerly of Harris Semiconductor) the CPU-16 is ready for silicon. Secondly, the CPU-32, somewhat redesigned from its original model expressed in discrete logic as befitted its intended incarnation as the Harris RTX-4000 (Harris demurred, and the CPU-32 is back in the hands of Epsilon Lyra), is in silicon samples.

Every university teaching computer instruction architecture should probably have its very own 32-bit writeable instruction set dual-stack Forth engine. Although about three-quarters of the microcode instruction pages in the silicon CPU-32 are now mask ROM supporting Forth, about one-quarter of the 512 pages are RAM and writeable on the fly. The CPU-32 also supports single-stepping microcode as well as reading and writing all major registers from the data bus during testing.

The CPU-32 executes Forth at about the same speed as the RTX-2000, but takes only one-half of the instruction fetch memory cycles. Very large memory space, high execution speed and the ability to enhance execution via customized instructions make the CPU-32 a unique solution to unique embedded control projects.

My current interest as an embedded systems designer in the CPU-32 is the possibility of using the CPU-32 to model custom fuzzy-logic microcontrollers. Since the CPU-32 has limited I/O facilities, and certainly no on-chip A/D or DAC, this would require memory-mapping the required hardware devices for testing, but what a prototype engine for a fuzzy instruction set this would make!

CPU-16/CPU-32 Contact Information:
Dr. Glen Haydon
Epsilon Lyra, Inc.
19500 Skyline Boulevard
Box 429 Star Route 2
La Honda, CA 94020-9726
(415) 747-0760

FRISC-4

This is Big Bertha. This is glorious, devil-may-care overkill in speed, size and beauty of architecture. This is the one that proves Forth-oriented microprocessor design is still surging into the future.

FRISC-4 is lineal descendant of the SC32, the orthogonal 32-bit dual-stack Forth Reduced Instruction Set Computer mentioned earlier in this article. The Johns Hopkins University Applied Physics Lab team of John Hayes, Marty Fraeman, Robert Williams and Robert Henshaw have by no means been resting on their laurels since embedding the SC32 in a magnetometer built for the Swedish satellite Freja. Modest success, in space and in the marketplace, has enabled them to winkle out enough funding to continue the saga.

Like its antecedent, the FRISC-4 has

- a simple 32-bit non-multiplexed address and data bus;
- a one-clock-cycle subroutine call;
- zero-cycle possible subroutine return;
- two parallel 16-register circular stack caches;
- programmer-transparent stack overflow/underflow shift to/from main memory;
- register access to top four stack elements of both stacks;
- support for mixed-speed memory subsystems.

The FRISC-4 is, however, different from the FRISC-3 is several important respects. The changes have come as a result of feedback from users of the part, and are, in my view, positive changes for the better.

First of all, the FRISC-4 address units are bytes, not longwords as on the FRISC-3. This means that overall the FRISC-4 can address one-quarter the memory of the FRISC-3, but not to worry: 512 Mbyte maximum code space and 4 Gbyte maximum data space are probably enough for the average embedded systems design, don't you think? The payoff is that character-oriented applications are either algorithmically simpler than before, or cost less for SRAM (7 nS SRAM isn't penny candy), or both.

Four general purpose global registers, (useful for shuffling operands while bursting the bounds of the Forth stack metaphor), and a barrel shifter which enhances the speed and range of shift instructions have been added to the original hardware design .

New instruction set features include leading zero detect for floating point normalization, a 59-cycle floating-point add and a 67-cycle floating point multiply.

Best of all for the embedded systems programmer is the on-chip interrupt controller, with INT0* through INT6* and an INTACK line. It is gratifying that the designers of both the FRP 1600 and the FRISC-4 have expended the effort to deal in a professional fashion with that which was the bane of early Forth chips, interrupt service.

You know now how big and how beautiful. How fast? Marty is sure of 33MHz and is crossing his fingers for a 50MHz part. Do you understand what 50 million 32-bit Forth instructions per second means? This will surpass the 32-bit integer performance of a 50MHz 80486 by a wide margin.

As noted, memory fast enough for this kind of part is expensive. To bring design-in costs down from outer space and into the earthly realtime control marketplace, the FRISC-4 will have an on-chip cache sufficient to keep performance high while allowing the use of slower, more economical memory devices.

FRISC-4 Contact Information:
Martin Fraeman
Johns Hopkins University / Applied Physics Laboratory
Laurel, MD
phone: (301) 953-6000 x8360
email: mef@glinda.jhuapl.edu

Incidentally, the FRISC-4 is not to be confused with a Micron Technologies microprocessor optimized for floating point also called FRISC. JHU/APL has been calling their Forth chip designs FRISC since 1986, but as we said, these chips have been marketed as the SC32, so no conflict has ever emerged in the marketplace.

Bibiliography

Footsteps in an Empty Valley, Dr. C. H. Ting, 1988, Offete Enterprises, 1302 South B Street, San Mateo, CA 94402, Tel. (415) 574-8250. The definitive work on the Novix 4000.

"Forth Machines", Jack Woehr, Embedded Systems Programming Vol. 3, No. 11, November, 1990. A survey of Forth engines.

"Forth stirs passion among its followers", Robert Bellinger, Electronic Engineering Times, July 6, 1992. Contains chart of recent very-high-end Forth projects, noteably in spacecraft. This article has helped stimulate the already-noticeable renaissance of interest in Forth in the general embedded control field. Overall well-written and fair, the article contains what is destined to be a long-remembered blooper, informing the reader that Forth words obtain input from the parameter stack and leave behind the results of an operation on the return stack. Well, we've all coded words like that at least once!

More on Forth Engines, Vols. I - XIV, Dr. C. H. Ting, Editor, Offete Enterprises. Periodical sampler of code and happenings in the world of Forth engines.

The Silicon Palimpsest by Charles Johnsen and David L. Fox, 1991 Proceedings of the SIG Forth of the ACM. Describes a "Mutable Instruction Set Computer".

"Space Application of SC32 Forth Chip", Silicon Composers, FORTH Dimensions XIV/1, May-June 1992. Illustrates high-end design-in of the highest-end Forth chip currently commercially available.

Stack Machines: the New Wave by Philip Koopman, Ph.D., John Wiley, 1990, ISBN 0470214678. Theory of the dual-stack CPU architecture inspired by Forth written by the then- Senior Research Scientist of the Harris Semiconductor RTX2000 project.

"Writeable Instruction Set Computers", Jack Woehr, Dr. Dobb's Journal #184, January, 1992. Describes the WISC Technologies CPU-16 with illustrative code example.

To reach Jack, use his Real Time BBS at (303) 278-0364. He is also a regular on GEnie's Forth Roundtable.

Thanks Jack, this was a great review of Forth Engines. Really makes me wonder why people keep pushing the xx86 CPU's. We can also see how much fun and excitment the Forth Conference was. Wish I had been there! Bill Kibler.

Real Computing

By Rick Rodman

32-Bit Systems

All Readers

Introduction to UNIX

Introduction to Unix, Minix, Coherent, and others

The operating system Unix began life as a single user word processing system inspired by Multics. Multics is the pioneer secure multi-user operating system, still in use in some installations. Unix was a spare time hack of Ken Thompson and Dennis Ritchie at AT&T Bell Labs. Dennis Ritchie invented the C language as a portable form of PDP-11 assembler, with concepts borrowed from PL/I and miscellaneous other languages, to allow Unix to be ported to other available minicomputers.

All of this happened about the same time as the development of CP/M in the mid-seventies. But while CP/M was made available commercially and became the basic software of most microcomputers, Unix was distributed to universities as an operating system concepts teaching vehicle. When commercial versions became available, they were horrendously expensive.

Originally, Unix was small and simple, but once released into academia, layers accreted on it in profusion. These additional masses of code caused various "versions" to diverge from one another, most notably the Berkeley version and the "true" AT&T version, more or less tolerated by AT&T for years before they started claiming they knew what they were doing all along.

Still, because most college-educated programmers worked with variants of Unix, most educational operating systems look like clones of Unix. Unix included some concepts which were revolutionary at the time, most notably a single vision of files and devices as being streams of bytes. I'm not sure this concept originated in Unix, however, because CP/M has the same concept embodied, most visibly, in the PIP program, although not in the OS itself.

The institutional bias of teaching generations of programmers with Unix has resulted in the ossification of its crudities such as *ls*, *mv* and *cp*, and its terrible case-sensitivity. Also, the C language has been frozen, warts and all. Dennis Ritchie did some things right - for example, the concept of everything being functions; but some things are really bad, like the awful C preprocessor, ternary operators, and other syntactical nastiness. In their favor it has to be argued they had no idea that people would still be using these tools sixteen years later. They had a job to do; they weren't Designing for the Ages.

Enough historical background. Any full version of Unix today is far too bloated to use for any educational purpose, so various versions have been written to replace it in this role. These serve various different markets, as it were.

Minix is still the best for those who want to work on the internals of the operating system itself. Andy Tanenbaum's book is very well-written and describes most of the concepts used in Minix, although it deals very sparingly with concepts not used. It also includes all of the source code of Minix version 1.1. The Minix 1.5 distribution includes installation instructions, user manuals, and complete source for Minix 1.5.10. Minix is a full-featured operating system. Version 1.6 may be released sometime soon; the Amoeba networking stuff included in version 1.5 is being deleted. Minix is available for the PC, the Mac, the Amiga, and the Atari ST. Since you get source, you can port it to any other random collection of hardware. That's how it came to be the OS used on the PC-532.

Coherent is Mark Williams' vision of a smaller, cleaner Unix. It's a mature, stable operating system which was originally offered for Z-8000-based S-100 systems sold by Ithaca Intersystems. From there, it moved to the PC. You don't get any source code, but if your goal is to *use* the system rather than tinker with it, Coherent will give you good performance at a lower price.

Linux is Linus Torvalds' clone of Unix for the 386 processor. Other CPUs need not apply. Getting a copy of the source is difficult; while the OS is apparently fairly stable at this point, this is experimenter's territory. Another OS not for the faint of heart is BSD386. This is a version of Unix from Berkeley with AT&T-copyrighted code expunged. Again, it only works on a 386. From what I've heard, this code is not very stable yet.

Another interesting Unix clone is Microware's OS/9, which runs on various 6809 and 68000-based platforms. It packs most of the features of Unix into a computer as small as Radio Shack's Color Computer. The problem with OS/9 is its unbundling into various high-priced packages - for example, no C compiler is included with it - so it's too expensive for most experimenters.

You can get various versions of real Unix for the PC, like Esix, Solaris, and so on. These cost more than you can afford,

require a bigger hard disk and more RAM than you have, and don't work with your hard disk controller or any of the peripherals you have. If you want X Window, it doesn't work with your video board, monitor, or mouse. Once you've bought a whole new computer to run it, you'll spend most of your time either trying to get the printer to work (ultimately unsuccessfully), adding another terminal, or fixing trashed filesystems. When the system is working, you can build your sense of achievement with trivial programs to convert the third letter of each word in a file to uppercase. Running Unix with Motif or Open Look is the most direct and expensive way to take a 40 MIPS RISC and make it perform like a Commodore 64.

The future of all of these products is unclear, including Unix itself. But that can be said of any computer product. Pick what makes sense for you, and keep your files in a form you can easily convert, such as MS-DOS 720K. My preference is Minix, because it supports several systems and includes source code. Your mileage may vary.

Ideally, I'd like a simpler, smaller, more portable OS than Unix. I'd like something that I could embed in a little Z-80 SBC to control my X-10 modules. But it has to use a language with expressive power and readability comparable to C, and allowing for heavy commenting - because after all, the writing of comments is the primary activity of a programmer.

Unix and C went quite a ways, but there's still a longer ways to go. The essence of clean design is the elimination of special cases. For this reason, operating systems and languages of the future will be less complex than Unix and C, not more complex.

CD-ROMs

Imagine that you are a collector of old coins. Every once in a while you have found an old dime in your change, or been able to scrimp and save to buy an old quarter or half dollar. Then, one day, you come across a man at a yard sale with a shoebox full of Franklin and Walking Liberty halves, silver dimes, and rare nickels and pennies. I mean, it's packed right to the top - it would take months just to examine each one. You look up at the man, pointing at the box, and he says: "You can't buy just that one, you have to take all *ten*. And I want *two dollars* for the lot."

That's about the way I felt when I first started looking through the Source CD-ROM from Walnut Creek CD-ROM. This disk contains, for example, all of volumes 1 to 27 of Usenet's comp.sources.misc archive; volumes 2, 8, and 89 to 91 of comp.sources.amiga, volumes 1 to 5 and 90 of comp.sources.atari_st, plus tons of Unix, Sun and MS-DOS code. There is so much source, it boggles the mind. Graphics - languages - BBSs - editors - you name it. MGR is here. MINT is here. Snobol. Several LISPs. Assemblers. PBM. Why, it's too much to assimilate! Arghh!

The price is about $30. If you are a software nut, or just a collector, and you have access to a CD-ROM drive, it's time to run and not walk.

What do we mean by "Real Computing"?

On this auspicious occasion of *TCJ*'s Tenth Anniversary, it was deemed appropriate that we finally define exactly what we mean by "Real Computing". This is a very nebulous phrase, and every computer-person probably has his own personal feeling about it. Originally I intended the phrase "real computing" to be computing in a similar sense to what we software folks refer to as "real programming". But that, too, is fairly loosely defined.

When you think about it, it isn't the *computer* that makes the *computing* real. The computing is what is done on the computer. So, what would make the computing "real" is the difficulty of the computation, such as number-crunching, Fast Fourier Transforms, flow analysis, et al.; or the constraints under which the computing is performed, such as time limitations or reliability requirements. Going beyond that, there's a complexity requirement. People who have programmed video games have impressed me with the stringent time constraints under which they have to perform fairly significant manipulations.

In addition to all of this, it seems to me that a computer that is only able to do one thing at a time is just not being used efficiently. Real Computing means getting every ounce of performance out of the machine, and that just can't be done in single-tasking interactive mode. So, I have to conclude that a minimum platform for Real Computing is a machine with a multitasking operating system.

When you sit down at a multitasking machine, a different mindset slowly seeps in. You don't have to wait for formatting or copying to finish. You can continue editing while the compiles progress. This is a work environment that fully utilizes the machine and the human being as well.

Where can one find a multitasking environment for doing some real computing? Many are available. Some representative multitasking OSs for micros include AmigaDOS, Minix, Coherent, Unix itself, Concurrent DOS, and the venerable MP/M. Computer power is becoming very inexpensive. What a tragedy to hobble a 486 with a tricycle OS like MS-DOS.

As I said, these thoughts may help you to formulate in your own mind just what Real Computing means to you. *TCJ* is, after all, the magazine for people who want to get extra value out of their hardware - the maximum "bang for the buck" - doing more with less. Tightening and optimizing code is itself one of the purest forms of Real Programming and, yes, Real Computing.

This column started with the NS32 CPU series and soon moved to the most popular end-user NS32 computer system, the PC-532, which runs Minix and uses SCSI for I/O. From there we've broadened to cover many topics related to Minix, SCSI, and the NS32. Today Minix is on the large side of what an experimenter's OS should be, and perhaps this is too broad a focus for a single column. I still hope

Continued on page 29

TCJ Center Fold

Special Feature

All Users

IMSAI CPA

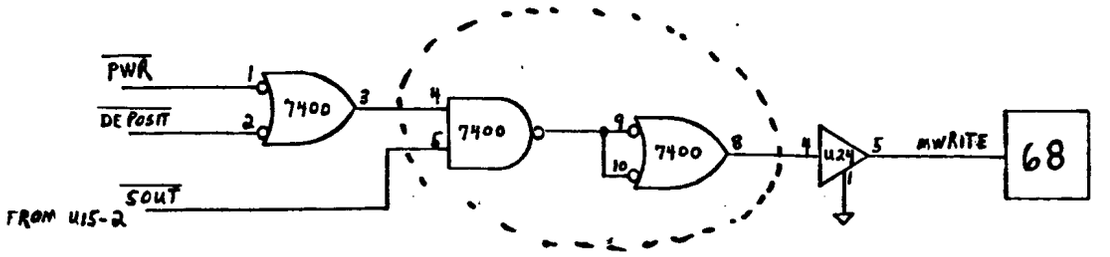
The IMSAI CPA is the control panel assembly used on IMSAI's early computers to control and enter data by hand. These units are the prize of many collectors. For S-100 debugging and initial system work, the CPA unit has no equal.

Using boards designed for the front panel, may not work properly when installed in systems without front panels. The MPU-A board presented in issue #59 showed changes to replace the MWRT signal. The front panel normally provides the MWRT signal (Memory Write) and a few select pullup devices (resistors to 5V that guarantee correct operation).

The Front panel provides single stepping options, the ability to enter data into memory locations by entering the data on the toggle switches, and monitoring overall operation through the display of status, data, and address signals with LED's.

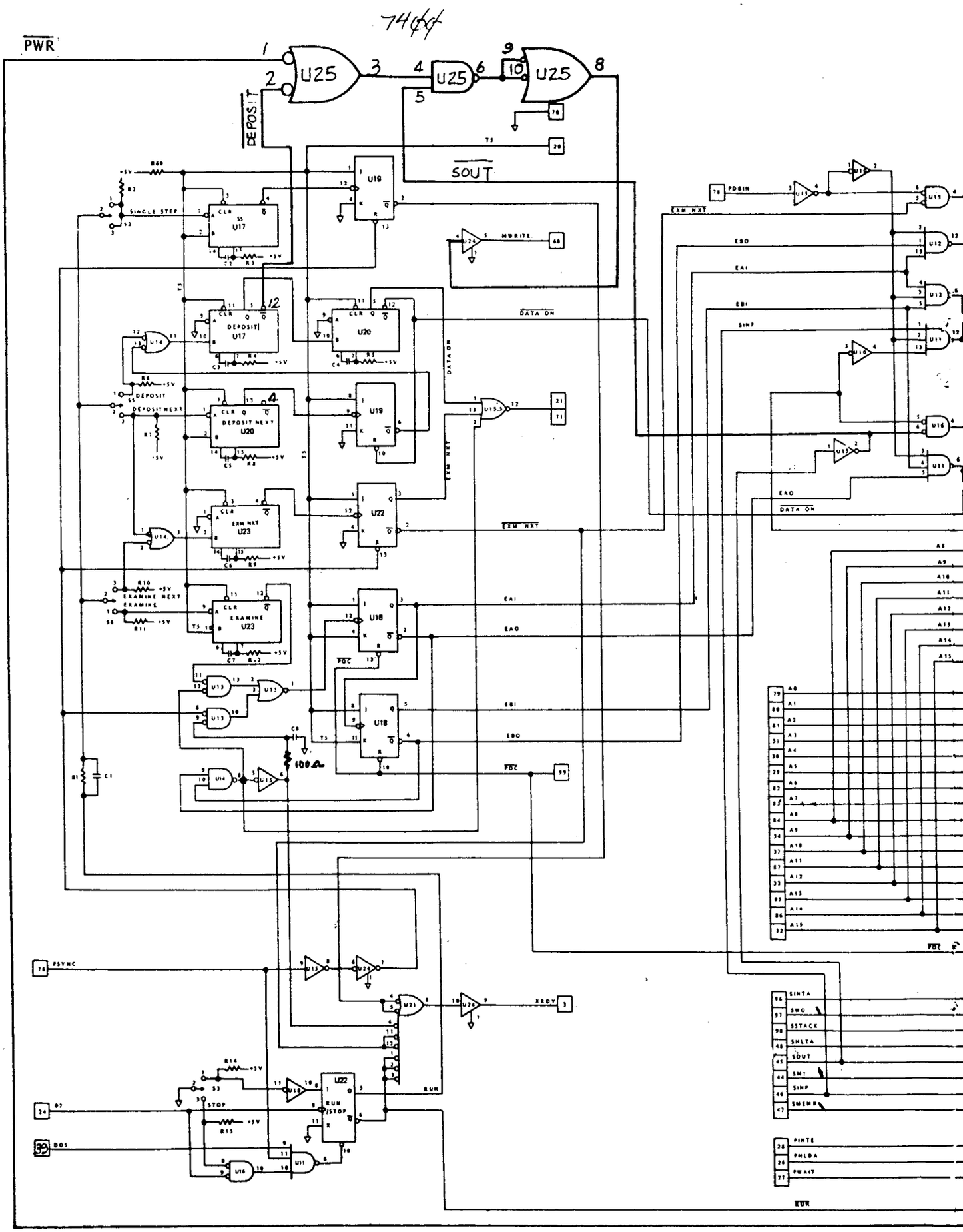
The last drawing (on page 28 of the center fold) is an example of the active portion of the S-100 bus. Higher quality mother boards will have active termination of the BUS signals. Active termination means a separate circuit and power supply which is used to provide a control voltage to the BUS lines. Unterminated BUS signals will have excessive noise and cause false operations. Higher speed operation is not possible without proper termination of BUS signals.

CP-A FRONT PANEL FOR REVISIONS 3 & 4
FOR USE WITH NON-IMSAI MEMORY BOARDS

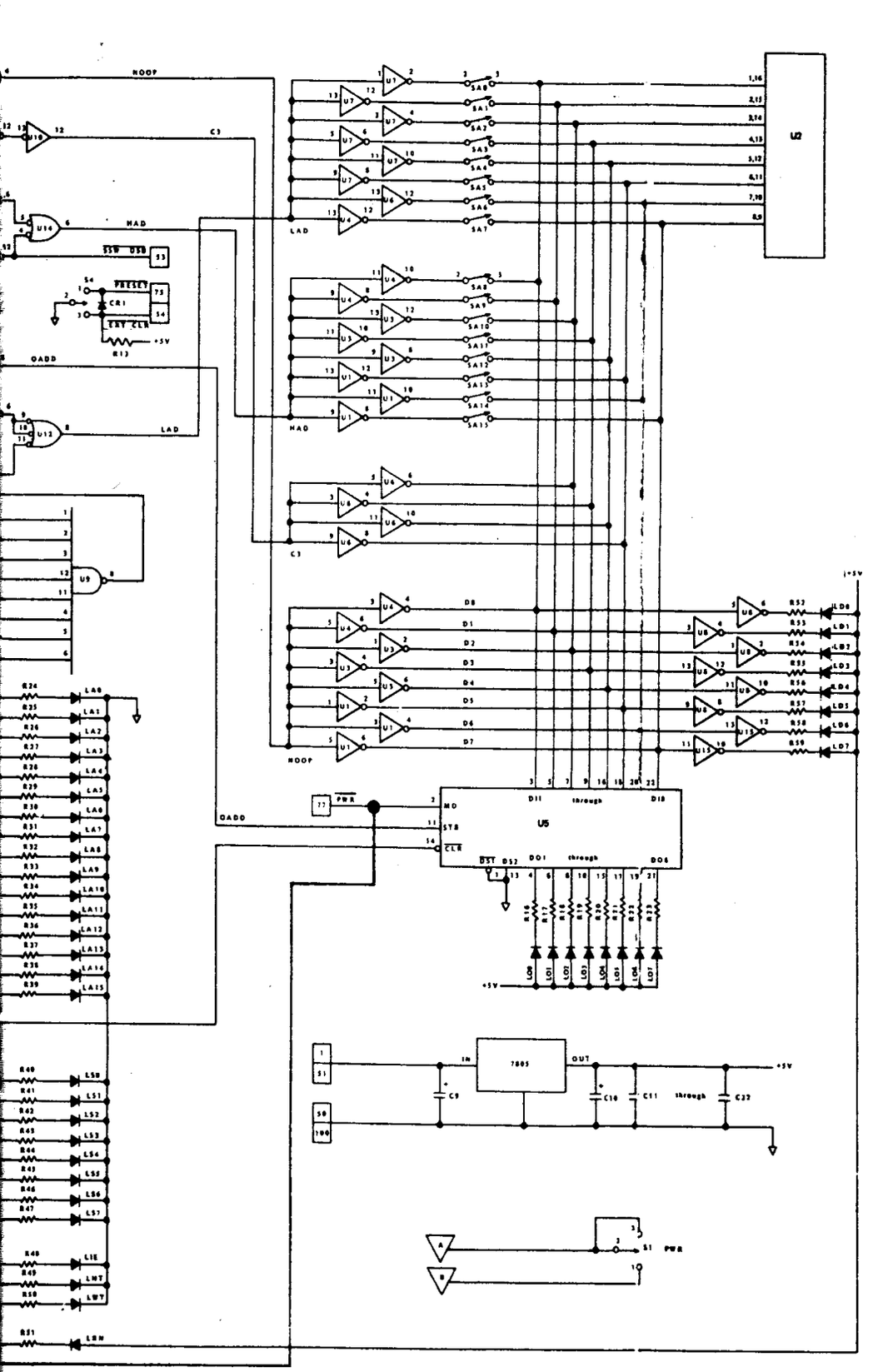


February 3, 1976

This modification should be made to your front panel board using two additional sections of the 7400 that is located directly above S-2 and S-3. This will keep your unit from writing into memory during an output.



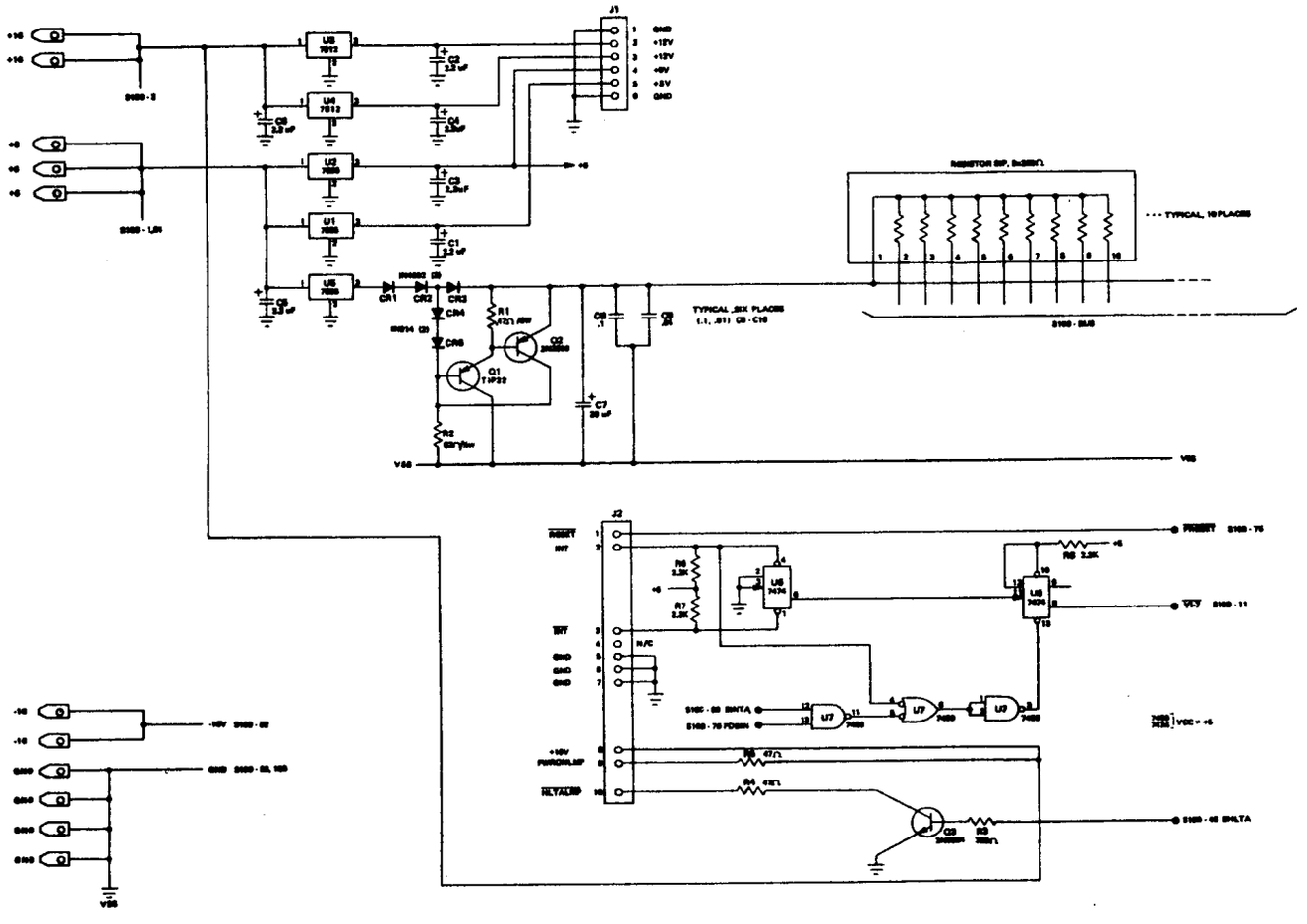
CPA 4833 2093



U1 U3 U4 U5 U7	7406
U2	DATA HEADER SOCKET
U6	8212
U8 U15	74LS04
U9	74LS30
U10	7404
U11	74LS10
U12	7410
U13 U16	7402
U14, U25	7400
U15.5	7427
U17 U20 U23	74123
U18 U19 U22	74107
U21	7430
U24	8T97
IC1 C2 C5 C6 C7 C11 thru C22	.1mF
C3 C8	.001mF
C4	.01mF
C9 C10	33mF
CR1	1N914
LD0 thru LD7 LA0 thru LA15 LD0 thru LD7 LS0 thru LS7 LIE LHT LRN LWT	LED'S
R1	270K¼W
R2 R6 R7 R10 R11 R13 thru R15 R60	1K¼W
R3 R4 R5 R8 R9 R12	47K¼W
R16 thru R59	220 ¼W

IMS Associates Inc.
SCHEMATIC DIAGRAM
CPA REV 4 2/76

2/27/76



REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

U1	7805	Q1	TIP 32
U2 * SEE NOTE		Q2	2N3906
U3 *	7812	Q3	2N3904
U4		CR1	1N4002
U5 *	7805	CR2	
U6	7474	CR3	
U7	7400	CR4	1N914
C1 - C8 * SEE NOTE		CR5	
C1	2.2 uF	J1	6 PIN CONNECTOR
C2		J2	10 PIN CONNECTOR
C3		R1	.47 2W
C4		R2	82 ¼W
C5		R3	220 ¼W
C6		R4	47 ¼W
C7	33 uF	R5	
C8	0.1 uF	R6	2.2K ¼W
C10		R7	
C12		R8	
C14			
C16			
C18			
C9	.01 uF		
C11			
C13			
C15			
C17			
C19			

*NOTE:
FOR PCS 80/15: REGULATORS U2, U5 CAPS C3, C5
FOR PCS 80/30, 34:
REGULATORS U2, U3, U5 CAPS C2, C3, C5, C6
FOR PCS 80/35:
ALL REGULATORS AND 2.2 uF CAPACITORS

TOLERANCES UNLESS OTHERWISE SPECIFIED FRACTIONS DEC. ANGLES ± ± ±	© 1977 IMSAI MFG. CORP. SAN LEANDRO, CA. ALL RIGHTS RESERVED WORLDWIDE MADE IN U.S.A.	
APPROVALS / DATE	EXP 10 REV. 2.2 SCHEMATIC	
DRAWN		
CHECKED	SCALE	SIZE B / DRAWING NO.
	DO NOT SCALE DRAWING	SHEET

to get enough nuts-and-bolts coverage to help you to understand, port, or even write your own operating system. In their own way, Real Computing and *TCJ* are chronicles of dreams too - highly individual and esoteric dreams, perhaps - but lofty and worthy ones nonetheless.

Next time

Next time we'll have a report on the emerging JPEG graphics standard and the International JPEG Group free JPEG source code. Keep your cards and letters coming. Tell your friends - *TCJ* is the magazine to maximize your MIPS!

Places to read more

Regarding the history and development of Unix: The Bell System Technical Journal, vol. 57, number 6, part 2, July/ August 1978; to a lesser degree, AT&T Bell Laboratories Technical Journal, vol. 63 number 8 part 2, October 1984.

Operating Systems Design and Implementation, Andrew S. Tanenbaum, Prentice-Hall, 1987.

Minix 1.5, software and reference manual, Andrew S. Tanenbaum, Prentice-Hall, 1991.

Where to call or write

BBS: +1 703 330 9049 (eves)

Walnut Creek CD-ROM +1 800 786 9907 or +1 510 947 5996
1547 Palos Verdes, Suite 260
Walnut Creek, CA 94596-2228

Mircoware Systems Corporation
1900 N.W. 114th Street
Des Moines, IA 50322
(515) 224-1929

Comming Next Issues

A new column on CP/M and other operating systems for eight bit systems.

The first set of articles will review CP/M organization and begining. The next articles will cover the utilities and their operation. Later articles will describe the BIOS and BDOS functions, leading to projects that add and change current BIOS functionality.

Later articles will cover other operating system such as: FLEX, SKDOS, OS-9, RP/M, CP/M68K and CPM86.

The first few articles are currently nearing completion. If you would like to cover one or more of the later topics, please contact TCJ.

Sending Articles to *TCJ*

Send your articles and letters of inquiry to:

The Computer Journal
P.O. Box 535
Lincoln, CA 95648

The editorial policy is to seek articles that can enhance and educate our readers. Letters of interest will be printed in our Reader To Reader section on a space and topic consideration. Material is typically printed "as is", however *TCJ* does reserve the right to reject or modify (by omitting) portions of letters or articles deemed unfit for publication. Any letters received by *TCJ* or it's technical editors may be printed or included within an article unless **YOU** indicate otherwise. Your name and city/state only will be used unless **YOU** indicate that you desire to have your full name and address included in references or letters printed.

Major letters and minor articles are accepted on floppy disk or by network services and will aid in getting your letter published "as is." *TCJ* does NOT return disks and material unless suitable and appropriate return mailers and postage is provided.

Floppy disk and word processing formats support by *TCJ*, are 3.5, 5.25, and 8 inch disk formats. Several CP/M to PCDOS conversion programs are used to transfer data for editing under WordStar with final output under PageMaker 4. Please do not use embedded punctuations in file names which can prevent reading by transfer programs. WordStar 7 can read and convert most other word processing programs output, but providing at least one ASCII file is recommended. Use of GENIE (as B.Kibler) and CompuServe (ID: 71563,2243) is the preferred method of sending information and articles to *TCJ*. Please ZIP files with a READ.ME, your article and an ASCII text version included.

TCJ is currently looking for articles that show our readers how you are still using older systems. Those systems can be anything except PC clone machines (machines like MBC 550 which are NOT 100% compatible are OK!). Your article should be written as if you are talking among friends and recounting your experiences. Please make it clear that "YOU" did this, and "I" had these problems, which "I" was able to resolve using these steps and techniques. The readers level of knowledge ranges from beginner to advanced. All references should provide a brief review of information to assist readers in determining the importance of the reference in relation to their own needs.

Special Feature

Programming

FORTH Support

Debugging Forth

by Walter J. Rottenkolber

So you got Forth on your system, figured out how the editor worked, and finally wrote your first long Forth program. And now, instead of a hoped for 'ok', your screen dissolved into fireworks, the keyboard froze, or you got a cryptic error message. What do you do?

KISS

Debugging code is a part of programming life. Forth can make the process less painful, but to do so, the debugging process must start when you write the source code. The trick, as all the Forth experts keep repeating, is to make Forth words short. Aim for no more than three lines of code. The use of screens at least reminds you of a 16 line maximum, but the habits learned coding in other languages die hard. Even after working with Forth for a while, I need to remind myself to keep it small and simple.

With Forth, the best method is to compile each screen, or even each Word, as it is written, and then test it. Ideally, by the time you reach the end of coding, the program should be debugged and ready to run. And you can do all this without first compiling the entire program, setting breakpoints, or using a complicated debugger that needs a workstation to run.

Charles Moore invented Forth not as a language, but as a productivity tool, a program development environment.

In a recent article on Shellsort (TCJ #57), I wrote the following code as a sort engine:

```
: SHELL ( -- )
        SETGAP BEGIN DECGAP
          ITEMS @ GAP @ DO
          I DUP S@ SV !
        BEGIN
          DUP GAP @ - DUP 0< NOT SWAP
          S@ SV @ > AND
        WHILE
          DUP GAP @ - TUCK S@ SWAP S!
        REPEAT
          SV @ SWAP S! LOOP
          GAP @ 2 < UNTIL ;
```

I was rather proud to have squeezed it into a single screen.

Today I would have written this code as:

```
: SHLMATCH? ( i -- i f )   \ f= true = no match
        DUP GAP @ - DUP 0< NOT
          \ Check if within array
        SWAP S@ SV @ > AND ;
          \ Compare array values

: SHLGETNXT ( i -- i' )
        DUP GAP @ - TUCK S@ SWAP S! ;

: SHLCOMPARE ( i -- i' ) \ i = array index
        BEGIN SHLMATCH?
        WHILE SHLGETNXT REPEAT ;

: PICKUPITEM ( i -- ) S@ SV ! ;
: INSERTITEM ( i -- ) SV @ SWAP S! ;

: SHLSHUTTLE ( -- )
        ITEMS @ GAP @ DO
        I DUP PICKUPITEM SHLCOMPARE
        INSERTITEM LOOP ;

: ENDGAP? ( -- f ) GAP @ 2 < ;

: SHELLSORT ( -- )
        SETGAP
      BEGIN DECGAP
        SHLSHUTTLE ENDGAP?
      UNTIL ;
```

Ok, I'll admit that it is a bit extreme and doesn't look simpler, but that's just because of unfamiliarity. The first example is Forth written like Pascal or 'C'. Here you take the pseudocode that outlines the functions and then transform it into a giant blob of working code. It is typical of the code you write before the big 'Aha!', when you finally understand what Charles Moore is driving at as the Forth Way. In Forth, the pseudocode becomes the Forth Words. These Words are then fleshed out in Forth 'one liners'. This begins the road to easy debugging.

For one thing you can more easily check if each little Word does what it is supposed to do before adding it to the next, more inclusive Word. This means more than just having the code compile, but discovering the conditions that may crash it. Because Words are self contained, you can probe them interactively or put them into test programs. By feeding SHLMATCH? with a wide range of index values you can discover the point at which it returns erroneous values or flags.

Conversely, it is easier to localize a problem to a specific bit of code if the Word is small. Consider how much easier it would be to find an error in the code of SHLMATCH? than the same code in the original version of SHELL.

Lastly, by choosing names that describe the function, the code becomes self-documenting.

You will also find it easier to understand what the code is supposed to do, because like food, code is easier to swallow in bitesize chunks.

Stack Gymnastics

One of Charles Moore's commandments is 'Thou shalt not PICK and ROLL'. This is another way of saying that stack manipulation should be limited to the top three elements as PICK and ROLL

enable you to copy and move stack elements deeper than three.

The purpose of the stack is to feed arguments (data) to operators (Words) in the Forth CPU. The stack represents a form of relative addressing rather than the absolute addressing common to most CPU's. If you examine the opcodes for the Z80 CPU they all take three or less arguments. Ditto with Forth.

I've written Words in which the stack has jumped through hoops, and done backflips and cartwheels. Today it would challenge anyone, including myself, to explain what the stack is doing. Code maintenance is a good reason for an uncomplicated stack.

Complicated stack manipulation is also a sure sign that the Word is trying to do too much, is too large. Such a Word needs rethinking and breaking up into smaller Words.

Procedures in other languages tend to be large and have long parameter lists. They can't be converted literally into Forth or you will end up with a complicated Word. Instead, you must abstract and translate the idea and spirit of the procedure.

I'm not saying that the stack must be limited to three elements, but that any excess should be considered temporary storage. Loading the stack so that the data is in the proper order often eliminates the need for manipulating below three elements. If a Word needs to draw on a large amount of data, it's usually better to write a set of variables or an array to hold them. This will make it easier to track the data for debugging and will also result in a simpler, more intuitive Word.

If you absolutely, positively have to manipulate more than three elements on the stack, then write a set of well defined, well behaved, and well documented primitive stack operators to do the job. Don't try to stuff all the code into one Word.

Sintax, Speling, and the Ubiquitous?

Whenever the compiler finds a word it doesn't recognize, the standard error response is to repeat the word (or underline it) and end with a question mark. Sometimes it enters the editor with the cursor placed after the offending word.

In the beginning, correct obvious syntax and spelling errors as you write. Depending on the Loader to detect bad spelling is a time waster. The errors may not be of your making. Printed source code often contains errors. Smart word processors act as though they have a mind of their own.

Dropped spaces are common. Unlike BASIC, Forth requires Words, numbers and strings to be separated by spaces for the parser to recognize them. With experience you will spot the more obvious errors. The top line of many screens is a comment line for the index Word. So if you see something like -- \Shell#3 -- you will know that it should be -- \ Shell#3.

More subtle is the invisible lost space. In the Forth screen, lines are a construct, not a physical reality in memory. The last column of a line is actually adjacent to the first column of the next line. So if you write code to the end of one line, and then start a colon definition on the next line, the space to the left of the colon will not be there. It's rather startling and puzzling to see the compiler choke on the colon Word. This became such an annoying problem for me that I modified my editor so that the last column of a line is always a space.

A similar problem is losing the semicolon of a definition so that the parser runs into the next colon word. This becomes more tricky if the missing semicolon belongs to the last word definition on the previous screen. If the first Word definition crashes at Colon, always check the previous screen.

Some Forths are case sensitive so that CR, cr, Cr, and cR are interpreted as different Words. Give me a break! I can never remember all the possible versions of the same Word name. The Laxen and Perry Forth83 has the variable CAPS, which, when set ON, makes the interpreter case insensitive so that all four versions of CR are treated as the same. I've set it ON. Real programming languages are case insensitive.

Always keep in mind this fact, the problem may not be in the word or the definition at which the compiler stopped.

Spelling errors are usually obvious. Less so is changing the name of the function and forgetting to search forward to change the name where it is used. Breaking large programs into smaller more self-contained modules makes it easier to find the Word you want.

Remember that Forth has a one pass compiler. Words must be defined before they can be used.

The Word may not be in the vocabulary(s) in the scan list. This is especially true if your system uses ONLY/ALSO. First check on the vocabularies being used by running ORDER. If you know the Word exists, there is nothing for it but to use VOCS to list the vocabularies and search through them. Switch to the vocabulary by just naming it. Though you can use WORDS, I prefer SEE as it is quicker. You add the vocabulary to the search either by just naming the Vocname, or adding it to the list by Vocname ALSO.

Less obvious is the changing Vocabulary or number base. It isn't that the compiler has failed, but when an error pops you out of the compiler, it stops at the last vocabulary or number base used.

I like to stay in Decimal mode, but sometimes switch to Hex if I have a complicated address or I/O code problem. If the error happens while in Hex mode, it stays there. However, you may not be aware of it, and so if you switch to another screen that requires decimal, the parser gets lost.

One tipoff of a number base change are the line numbers at the edge of the screen display. If you see letters, you're in hex.

The same happens if the vocabulary search order gets changed and no longer searches the one with the Word in it. Run ORDER to check this.

I had enough problems with this that I now have the status line in the editor display the number base and current vocabulary.

The Stack

Another error is for the compiler to stop after the definition of a word with the error message "Stack Changed" or "Conditionals Wrong". This often occurs in definitions with branches, loops and case statements when you leave out one of the Words in the definitions.

Most commonly it occurs in nested IF THEN or IF ELSE THEN constructs when you have too few THEN's. If I get carried away with such a branching mess, I now count IFs and make certain that they are balanced with an equal number of THENs. Better still, use the CASE statement if a single value selects one of many branches.

"Stack Underflow" is an obvious error. Somewhere a Word either didn't leave a value on the data stack or took off one too many.

More subtle is having too many items on the stack after the Word has run. I now make the effort after testing a Word to use Show Stack (.S) to detect this problem. Some Forths give a "Stack Overflow" message if a runaway loop fills up the data stack.

The Forgotten @

Most procedural languages provide the value of a variable simply by naming it. Data is stored in the variable by the assignment method introduced in Fortran and formalized in Algol 60. This is written like an algebraic equation that reads backwards, from right to left.

The paradigm for Forth, however, is assembler, so the movement of data generally makes explicit use of pointers (addresses of data). The exception is constant which fetches a value simply by naming it. Between that and the ingrained habits from other languages, forgetting to fetch (@) data from a variable is a commonplace error. This can lead to interesting times, but there are clues that point to this.

If a variable value doesn't change as it should or does so erratically, check first for a Forgotten @ before tearing up otherwise good code.

The addresses of variables are usually larger than most index values. In a DO LOOP this can cause either premature shut off or loop runaway depending on the index (initial or limit) effected.

Addresses of variables don't change and are non-zero, so a variable that holds a flag will always appear True. Finding IF ELSE THEN branches that don't ELSE is a good sign that the flag value is not being used. The same goes for BEGIN UNTIL loops that loop only once, and BEGIN WHILE REPEAT loops that never stop and lock up the computer.

DO +LOOPs

DO LOOPs that run the the positive direction stop at one less than the limit index. Since Forth generally has standardized counts as beginning with zero, the count comes out correctly. However, if you do a +LOOP with a negative number in order to count down, the loop stops AT the limit index, that is, one more than the positive direction. Always double check +LOOPs going in a negative direction for the proper count, and adjust the indexes accordingly.

>R and R>

Using the return stack to temporarily hold data can solve a sticky parameter stack problem. But, whenever you see a >R, begin the search for the R>. It must occur before the Semicolon or an EXIT. The return stack holds the address Next will return to when the present word is done. An extra value on the return stack will most likely cause a system crash. Removal of an address will cause Next to prematurely exit the calling Word.

If you store a value on the return stack inside a DO LOOP be certain that the value is returned to the data stack before you reach LOOP or LEAVE. DO LOOPs store the index values on the return stack and take them off to process them. So an extraneous value on the return stack will cause the DO LOOP to do strange things. This also goes for fetching an index value within the loop, e.g. >R I R> will crash.

Dangerous too is code like:

```
BEGIN >R (code) WHILE R> (more-code) REPEAT
```

as this runs just fine until you leave the loop. If there is no R> before the semicolon, the Word will crash.

Snapshots

One method to discover what happening is to open up the code by inserting Snapshots. These are Words that will display the stack or variables as the code runs. I've used Words such as .S, ?, or the various PEEKS (see screens). By arranging judicious CR's, data from within loops can be displayed in a neat line format. If the display moves too fast, inserting a WAITKEY or PAUSEKEY will halt the Word so you can read the data. I write these Words in lowercase to distinguish them from the uppercase program code. Try to place them at the end of the line where they can be more easily seen and removed.

Dummy Words

If you test Words interactively, Words that activate disk or port input and output could inadvertantly crash the system. So set up a screen of dummy Words in which DISKWRT is a NOOP, and PORTs I/O data to a variable. This will let you simulate a working program and check out every Word except the few hardware dependent ones.

Deep Probe

So far, just looking at the code or thinking about how the program fails can provide a clue as to what the problem might be. But eventually you may need to probe deeper.

Most Forths have utility programs that provide tools for debugging.

The standard tools in the Laxen & Perry Forth 83 are VIEW, SEE, DUMP, DU and DEBUG.

In my system, I've replaced the standard screen editor with a modified version of F. van Duinen's excellent PDE v2.02. This editor allows you to work with screens as handily as a text editor does text. I highly recommend it even though I had to spend time replacing some IBMpc code with Kaypro compatible code.

It also contains a single-step debugger that works on uncompiled source code within the editor. It displays the stack similar to DEBUG. However, it cannot work with compiler words, such as IF THEN, or Words that change the return stack, so it is limited. I use it for a quick check of stack behavior.

VIEW allows you to automatically look up the source screen of the desired Word, provided the source file is online. You can check the stack comment or make certain the Word does what you want it to.

Use as -- VIEW wordname.

View screens are accessed by an integer in the Word header which points to the filename pointer in the VIEW-FILES array, and gives the number of the screen in which the Word is defined. To enter a file into the View system do:

n VIEWS filename

Where n is the view file location in the pointer array to the viewfile FCB. This should be done in screen #1 before the rest of the file is loaded. Up to 16 files can be put into the View array. In the Laxen and Perry Forth 83, locations one through four are taken by the Forth kernel and extensions. My system has n=0 left free for temporary View files, primarily files of work in progress.

The only downside to VIEW is that using it closes the file you are working on, and you have to OPEN it again. Another advantage of PDE is that you can tag the working file so that a few key strokes will restore it.

The FCB of the source file contains the view number which is then incorporated into the Word header. If the drive originally compiled into the FCB is not convenient, it can be changed without recompiling by VIEW-CHGDRV (see screens).

SEE is a high level disassembler. If the View files are not accessible, it will provide a clue as to the function of a Word. Since the Words are compiled, the result is not the same as source, particularly the appearance of branches and loops. With a little practice comparing source with disassembled code, you should be able to interpret it.

Use as -- SEE wordname.

DUMP is a memory dump similar in appearance to that found in your typical debugger. Sixteen bytes in Hex are displayed in a line on the left with printable characters displayed on the right.

Use as -- start-addr #bytes DUMP.

DU is a short dump which displays 64 bytes.

Use as -- start-addr DU.

DU leaves the next starting address on the stack, so that to see the next 64 bytes just type DU. (It also means that you have to drop the address when done).

The starting address of a Word or array may be found by using Tick (').

Remember that the Z80 uses the 'little endian' method of storing 16-bit integers. This means that the least significant byte is stored in memory lower than the most significant byte. For example, A067 will be seen in Dump as 67 A0. When searching memory for data you must know its size and location in the Forth Word. You will need a knowledge of Forth dictionary structure. That means an investment in the appropriate Forth reference texts and some study.

DEBUG is a powerful debugging tool. You set it up by typing:

DEBUG wordname

After you press Enter, nothing will seem to have happened. However, if the Word to debug is run, the debugger is entered. Each word in the parameter field is then displayed along with the contents of the stack. Pressing the spacebar or Enter will single step through the Word.

The actions of the Words also occurs. So if messages are typed, the screen quickly becomes cluttered. If the cursor is moved, the stack display will shift also. When KEY input is required, you must do so just as if the program were running normally. With practice you will be able to follow the debugging process though it is confusing at first.

By pressing 'C' the debugger will run continuously and allow you to fast forward through the word. Pressing any key puts you back into single step mode.

Pressing 'F' puts you into the Forth command line. You won't see an 'ok' until some action is taken, so at first nothing seems to be happening. This lets you change the stack contents, run another Word, or check a variable. Typing RESUME puts you back into the debugger at the point you left, but be certain that the stack is correct or you will crash. Use .S to check.

To quit the debugger press 'Q'. This unbugs the Word too and must be done before the debugger leaves the Word.

When you're in DEBUG, the extra code it uses will noticeably slow your system. This is normal.

I usually start debugging at the top, unless the behavior of the program provides a clue to the defective code section. Writing small Forth Words will let you follow the trail of offending Forth Words rapidly. (Keep the Reset button closeby). If the code were all like that in SHELL, dredging through it would be real work.

The debugging process in Forth consists primarily of checking that the stack has the correct values in the proper order, and that the Words provide the function you expected. If you don't know, VIEW is there.

For example, in Forth83 NOT is the one's complement and not the same as 0=. So I need to be aware that while TRUE (-1) will NOT to FALSE (0), other non-zero values won't and I must use 0=.

Converting figForth to Forth83 provides the ultimate object lesson in obscure code behavior.

I've learned that any hint of doubt must be double checked. VIEW and SEE provide valuable information at your fingertips. Forth's interactive mode makes it easy to try Words out. Use it.

Fuzzy Logic

Not for the first time have I written code like --

```
APPLE DUP @ (more code) !
```

and discovered the program to go astray. The debugger will quickly show you that the stack holds -- (var-addr value --). However, Store (!) requires the address on top, so when SWAP is added before Store, all is well.

Then take the case of --

```
A) B/BUF  TXTPTR @ < IF  WRITE-BUF ...
B) B/BUF  TXTPTR @ > IF  WRITE-BUF ...
C) None of the above
```

A) will store one byte past the end of the buffer. You lose one byte each write, and have a possible crash at the next Word in memory. B) will crash. So, C) is the answer.

You need something like:

```
B/BUF TXTPTR @ <= IF.
```

Confusing the scope of a count with that of an offset is the cause of this error. Counts go from 1 to n. Offsets go from 0 to n-1. As a result, when TXTPTR equals B/BUF, the pointer is actually one byte past the end of the buffer.

The problems caused by fuzzy logic are the most difficult to debug. Nothing is more frustrating than code that compiles, and then almost works. Taking the time to test Words with critical values is the only way to ensure correct function. Sorry to say, but at this point, having made mistakes (aka experience) is the best means to learn what fuzzy logic will do.

Writer learn by writing, programmers learn by programming.

Wrapping up

As you write larger programs, a few tricks help to make it easier.

Try to divide the program into functional modules. Then extract from them any primitive Words and data structures that are used globally. The remaining data and Words will then form selfcontained modules. The global code now provides a base on which the modules can rest. This means that you can work on one module without having to load the other modules.

If you are continuing work on a base of completed code, compile what is finished. Then add a dummy word such as TASK or use the Word MARK to mark the top of the dictionary. Save the system under a single letter file name. When you need to hit the reset button after the inevitable (for me anyway) crash, running the saved program will put you back to square one without having to reload the entire program. In addition, FORGETting TASK will only dump the latest work leaving the rest intact.

Homebrew Tools

Because Forth is so open, making simple but effective debugging tools is not only possible but encouraged.

PATCH redirects code accessing one Word to use that of another Word. This allows you to try modified Words without a complete reloading of code. UNPATCH restores the last Patched Word to its previous state.

TO> allows you to change the value of a constant without recompiling. Ordinarily it's bad form to change the value of a constant in running programs, but when debugging it may furnish the vital clue.

Use as -- n TO> const-name

.VIEW will list the files in the viewfile array so you know which positions are still open for use.
VIEW-CHGDRV will change the active drive of a view file if it has been compiled with a drive inconvenient for you. Though .VIEW shows the drive as a letter, in CP/M, the drive is actually designated by a number in the first byte of the FCB. A '0' means the default drive, a '1' is the 'A' drive, and so on up to '16' for drive 'P'. So you'll have to convert the drive letter to the number to use VIEW-CNGDRV. The filename position can be found by running .VIEW. Run SAVE-SYSTEM to make the changes permanent.

The next two Words repeat Forth Words from the interpreter command line. They complement the MANY, TIMES, and :: Words found in the Utility.blk. These Words allow for quickie tests on Words interactively.

RUNAGN Continuously repeats the interpreter command line code. Pause by pressing a key. Leave the loop by pressing Escape.

Use as -- command-line-code RUNAGN

STEPAGN Single-step repeat of interpreter command line code. Repeat by pressing a key. Leave the loop by pressing Escape.

Use as -- command-line-code STEPAGN

Conclusion

Obviously, no debugging tool will substitute for clear, reasoned thought. These tools only provide raw data. To interpret the data you need an awareness of how the program should behave, and what the data values should be. You can't detect abnormal behavior in code without first knowing the normal. Practice debugging on working code and follow the flow of data on and off the stack. The Forth source itself is a goldmine of code examples and ideas.

My best learning experiences come from modifying code to make it 'work better'. Wending your way through working code makes you think through the logic of the program, shows you how Words are linked to make it work, and encourages you to read your references on programming.

I've presented some of the methods and reasoning that I use to take the kinks out of my programs. As long as you use copies of code and have a reset button handy, making a mistake is no problem. If you have the time, set up code with deliberate errors. See what happens. Remember it for the time you have to deal with the real thing.

Accept the challenge and have fun.

-- THE END --- CODE FOllOWS --

```
\ Debugging Forth Screens          WJR23OCT92

          Debugging Forth
          Walter J. Rottenkolber
          Oct 1992

\ PATCH    WJR25OCT92

2VARIABLE SAVPATCH1
VARIABLE SAVPATCH2

: SAVEPATCH ( n -- )
          DUP 2@ SAVPATCH1 2! SAVPATCH2 ! ;


: PATCH ( -- ) (S newword oldword )
          ' ' DUP @ ['] : @ =
          IF >BODY DUP DUP SAVEPATCH -ROT !
              ['] EXIT SWAP 2+ !
          ELSE ." Patch Error" ABORT THEN ;

\ Redirects calls from oldword to newword

: UNPATCH ( -- ) \ Undoes the last Patch
  SAVPATCH1 2@ SAVPATCH2 @ 2! ;

\ Change View Drive    WJR25OCT92

: .VIEWFILE ( a i -- )
          DUP 2 .R ." = " 2* + @ DUP IF
                    >BODY .FILE
          ELSE DROP ." No File" THEN ;

: .VIEW ( -- ) \ Lists Files & Positions in View-File Array

          VIEW-FILES 16 0 DO
          DUP I CR .VIEWFILE
          LOOP DROP CR ;

: VIEW-CHGDRV ( drv# vwpos# -- )
  2* VIEW-FILES + @ >BODY C! ;
\ Changes drive for viewfile search.
\ drv# 0= defit., 1=A, etc.; vwpos# is position of file.


\ Snapshot Tools       WJR24OCT92

: BINARY ( -- ) 2 BASE ! ;
: PEEK     ( n -- n ) DUP . ;
: UPEEK ( u -- u ) DUP U. ;
: HPEEK ( u -- u ) HEX DUP U. DECIMAL ;
: BPEEK ( u -- u ) BINARY DUP U. DECIMAL ;

: WAITKEY ( -- ) KEY DROP ;
: PAUSEKEY ( -- ) KEY? IF WAITKEY WAITKEY THEN ;
```

```
\ RUNAGN  STEPAGN          WJR04NOV92

: ?ESC ( -- f ) KEY 27 = ;

: RUNAGN ( -- )
          KEY? IF ?ESC IF EXIT ELSE
          ?ESC IF EXIT THEN THEN THEN >IN OFF ;

\ Repeats Command Line. Pauses on key press.
\ ESCape exits.

: STEPAGN ( -- )
          ?ESC IF EXIT THEN >IN OFF ;
\ Repeats Command Line when key is pressed.
\ ESCape exits.

\ TO>      WJR04NOV92

: TO> ( n -- ) (S constant-name )
          ' >BODY ! ;
```

```
\ Screen 0
.\ Screen to Text Files                WJR04NOV92
\S
     Screen to Text Files
     Walter J. Rottenkolber
        Nov. 1992
  A utility to convert screen blocks to text files.
  Places screen number above line 0. Can be set up to
  remove all blank lines within screen or only multiple
  lines. A blank line separates screens.

\ Screen 1
\ Screen to Text Files                 WJR05NOV92
ONLY FORTH ALSO  FORTH DEFINITIONS
DECIMAL
2 9 THRU

\ Screen 2
\ Screen to Text Files                 WJR28OCT92
\ 1024 CONSTANT B/BUF  \ Uncomment if needed
\  64 CONSTANT C/L
\  16 CONSTANT L/SCR
  VARIABLE TXTBLK#
  VARIABLE TXTPTR
: PAD2 ( -- a ) PAD 100 + ;
: PAD3 ( -- a ) PAD2 B/BUF + 10 + ;
: INITS>T ( -- ) 0 TXTPTR ! 0 TXTBLK# ! ;
: WRTXT ( -- )
  1 MORE PAD3 TXTBLK# @ BUFFER
          B/BUF MOVE UPDATE ;

\ Screen 3
\ Screen to Text Files                 WJR05NOV92
: ?WRT-TBUF ( -- )
  B/BUF TXTPTR @ <= IF WRTXT
  1 TXTBLK# +! 0 TXTPTR ! THEN ;
: >TBUF ( c -- )
  ?WRT-TBUF PAD3 TXTPTR @ + C! 1 TXTPTR +! ;
: NEWLINE ( -- ) 13 >TBUF 10 >TBUF ;
: EOFMARK ( -- ) CONTROL Z >TBUF ;

\ Screen 4
\ Screen to Text Files                 WJR28OCT92
  VARIABLE 0LINE?  DEFER ?BL-LINE
: ?BL-LINE1 ( a i -- a i f )
  0LINE? @ OVER OR 0<> OVER 0<> 0LINE? ! ;
: ?BL-LINE0 ( a i -- a i f ) DUP 0<> ;
: (PUTLINE) ( a i -- )
  0 ?DO DUP C@ >TBUF 1+ LOOP DROP ;
: PUTLINE ( a i -- )
  ?BL-LINE IF (PUTLINE) NEWLINE ELSE
          2DROP THEN ;

\ Screen 5
\ Screen to Text Files                 WJR05NOV92
: .SCR# ( -- )
  " \ Screen " (PUTLINE)
  SCR @ (U.) (PUTLINE) ;
: 0LINE ( -- ) 0LINE? @ IF NEWLINE THEN ;
: 0LINSTUF ( cnt -- )
  IF 0LINE THEN .SCR# NEWLINE ;
\ Adds newline if previous line #15 and present line #0
\ contain text. Then prints Screen number before line #0.
```

```
\ Screen 6
\ Screen to Text Files                 WJR05NOV92
: GETLINE ( line# -- a i )
  C/L * PAD2 + C/L -TRAILING ;
: ?ABORT ( -- )
  KEY? IF KEY 27 = ABORT" Program Aborted" THEN ;
: (SCR>TXT) ( -- )
  0 GETLINE DUP 0LINSTUF PUTLINE
    \ Zero-line routine
  L/SCR 1 DO I GETLINE          \ Rest of screen
  PUTLINE ?ABORT LOOP ;
: GETSCRBLK ( blk# -- ) IN-BLOCK PAD2
          B/BUF MOVE ;

\ Screen 7
\ Screen to Text Files                 WJR28OCT92

: IN-CAPACITY ( -- n )
  IN-FILE @ [ DOS ] MAXREC# @ 1+ 0 8 UM/MOD NIP ;

: (S>T) ( n n -- )
  INITS>T DO I DUP SCR ! GETSCRBLK
  (SCR>TXT) LOOP
  EOFMARK WRTXT FLUSH ;

: ?S>TALL ( [n n] -- n n )
  DEPTH 2 < IF IN-CAPACITY 0 ELSE
  1+ SWAP THEN ;

\ Screen 8
\ Screen to Text Files                 WJR04NOV92

: FILE:FROM-TO ( -- ) (S from-filename to-filename )
  [ DOS ] FCB2 DUP !FCB FILE !
  FCB1 DUP !FCB DUP DELETE DROP
     \ Deletes existing to-file
  DUP MAKE-FILE IN-FILE !
  OPEN-FILE SWITCH OPEN-FILE ;

: S>T ( [from to] -- ) (S scr-filename txt-filename )
  FILE:FROM-TO ?S>TALL (S>T) ;

\S Use as --
  from-scr# to-scr# S>T scr-filename txt-filename ( or )
  S>T scr-filename txt-filename \ Converts entire scrfile
     Press ESCape to abort program.

\ Screen 9
\ Screen to Text Files                 WJR03NOV92

: 1BL-LINE ( -- ) \ Removes multiple blank lines
  ['] ?BL-LINE1 IS ?BL-LINE  0LINE? OFF ;

: 0BL-LINE ( -- ) \ Removes all blank lines between text
  ['] ?BL-LINE0 IS ?BL-LINE  0LINE? ON ;
  0BL-LINE \ Default is no blank lines
  .( Screen to Text screens loaded)
```

Regular Feature

Kaypro Support

ROM Modifications

Mr. Kaypro

By Charles B. Stafford

SIR KAYPRO

Wherein we wave the magic wand, and turn a mouse into a LION.

RESPONSES FROM READERS

Boy, am I HAPPY, not only did I get mail, but I now know what at least one reader wants to hear about !!!!

When, in the last TCJ, I asked for Reader response, I hoped for perhaps a single letter. I do like feedback, but was not too optimistic. My wildest hopes were realized, when I received not one but THREE letters, all with very useful information. One, in particular, has caused me to deviate from my original plan for this column. The gist of the letter was, that a pristine Kaypro II had been purchased for a song by one new to computers who had sung the song himself, was reasonably happy with the present capabilities of his prize, but had heard through the "grapevine" that certain "enhancements" could be performed by a beginner with basic manual dexterity. He had high hopes that he could be shown a relatively simple path to implement this "wizardry".

THE VISION

Based on this plea, it appeared that a step back was called for, to lay out the possibilities. Those of you that remember "Micro Cornucopia" from a prior lifetime, will recall that the subject was discussed in that publication at some length, and a great deal of what follows is loosely based on that research and those designs.

BEGINNINGS

In the beginning....

There were two early non-harddrive Kaypros, the Kaypro II and the Kaypro IV. The difference was that the II had only single sided double density drives (190k) but the IV had double sided double density drives (390k). One very interesting quirk was that, if you put a II boot disk in a IV and either powered-up or used the reset button to reboot, the IV would agree with you that it was only a II and would read only SSDD diskettes. Booted up as a IV, however, it could still read II diskettes as well as its own. It turns out that a II can be converted to a IV rather easily, and if that step is performed first, all the other projects we discuss would be applicable to either machine. My first choice, then, for an upgrade would be the II to IV conversion.

CHOICES

Eenie Meenie Mynie Mo.....

After the II to IV conversion, come several choices. The original floppy drive only machines ran at 2Mhz. For comparison, the IBM PC-XT runs at 4.77Mhz. The Kaypros seem as fast because they are character based, and don't have the video overhead (extra operating system software) to run each time they display a character. The system clock (oscillator), however, runs much faster and the frequency is "divided" to produce the 2Mhz signal for the processor. Both 4Mhz and 5Mhz signals are available on the motherboard though, and substituting one of those for the 2Mhz signal makes the Kaypro much more responsive, although my daughter says that LADDERS becomes "real tough". There are both 4Mhz modifications and 5Mhz modifications as well as one with a switch to slow down to stock again. Actually, there is a 7Mhz signal available also, but the display is the limiting factor, and since it cannot "keep up", the processor running at 7Mhz has to wait for it, so there isn't any real reason to use 7Mhz.

So much for the speed freaks. For those who value memory above all else, 256k of ram can be installed. Only the first 64k can be used by applications programs, but there are drivers to use the remaining room as a print spooler or a "ram disk". This is a little more advanced as projects go, but anyone who is relatively competent at soldering can handle it IF they are careful.

For those who value disk drive space, up to 4 floppy drives can be installed as large as 800k each. This modification requires a new monitor ROM and the addition of drive select decoding hardware (another IC) but is not difficult. I suppose I should mention that the drives must be procured as well, either at a swap meet, a used computer store, or new.

For those who REALLY value disk drive space, up to TWO 55MB hard-drives can be installed. Again not difficult, but rather more expensive than the other modifications already mentioned.

For those for whom convenience is paramount, moving the reset button, as discussed two issues back, and likewise moving the brightness control can be done by careful beginners at very little expense.

THE RESULT

The end result, could very well be a Kaypro with 256k of RAM, 4 floppy drives ranging from 390K to 800k, both

3.5" and 5.25", and 110 megabytes of harddrive space all running at 5 Mhz, and all packed in the same aluminum case it came in.

TODAY'S PROJECT

Since moving the reset button,(our least risky and most cost effective project) was covered two issues back (#58) We'll proceed to the logical successor, the II to IV conversion. See page 39 for a drawing of the jumpers and parts discussed.

Before we start, we need to ascertain whether the patient is really a K-II or a K-IV in K-II clothing. Remove the hood by removing the 10 screws holding it on, and lift it off. Please disconnect the power FIRST. If U-47 is identified as an 81-149, the patient is a real K-II, if U-47 is identified as an 81-232, you have a K-IV in K-II clothing. If U-47 is not identified, look for a mother board assembly number on the right corner closest the front of the computer. an assembly number of 81-240 indicates a K-IV masquerading as a K-II. If you have a real K-II then proceed, if it's actually a K-IV in "drag" skip to the replacement of the drives.

There are two parts to this conversion, first the board needs to be modified to use a 2732 EPROM instead of the 2716 that was stock. This is to allow the extra code for side select. Then, second, the side select line must be physically connected to the floppy drive cable. The cleanest way to do this is with extra sockets for those ICs that are socketed. You'll need 5 IC sockets, 1 24 pin, 1 16 pin, 1 14 pin, 1 20 pin, and 1 40 pin. By using these sockets we can avoid soldering on the board itself, and still produce a neat workmanlike result.

THE EPROM MOD

Remove U47 using a very small screwdriver to pry it out.

Bend pin 21 of the 24 pin socket out so that it will not go into the U47 socket when inserted like an EPROM.

At this point, a small digression is necessary. Those of you who are already certified as WIZARDs can skip this, but for those of us who are not, a small review may prevent a large error. The pins on an IC are numbered COUNTER-CLOCKWISE from the notch or dimple, when viewed from the top. For those who grew up with digital watches, if you look at an IC, (right side up, of course) with the notch/dimple on the left, pin 1 will also be on the left, on the side closest to you.

Solder one end of a two inch jumper wire to this bent out pin.

Solder the other end of the jumper to pin 2 of the 20 pin socket.

Bend out pin 1 on the 16 pin socket so that it will not go into the U 60 socket when inserted like an IC.

Solder one end of a one inch jumper to pin 1 of the 16 pin socket and the other end of the jumper to pin 8 of the 16 pin socket.

NOW

Remove U 59, and U 60, using the screwdriver as above, and:

1. Insert the 24 pin socket into the socket at U47, keep pin 21 bent out

2. Insert the 20 pin socket into the socket at U 59, all the pins must go into the socket

3. Insert the 16 pin socket into the socket at U 60, pin 1 must remain bent out

What we've done is to add a select circuit to pick up the extra 2k in the larger ROM, and connect address line A11 to pin 21 of the larger ROM. The original ROM will NOT work now, but an original Kaypro-IV ROM or one of the aftermarket ROMs will.

THE SELECT LINE

"This will be much simpler than the last procedure" he said with an evil grin.

Remove U 72 the PIO and U 73 from their sockets using that same screwdriver.

Bend out pin 13 on the 40 pin socket and pins 5 & 6 on the remaining 14 pin socket.

Solder one end of a two inch jumper wire to pin 13 of the 40 pin socket and the other end of the jumper to pin 5 of the 14 pin socket.

Solder one end of a four inch jumper to pin 6 of the 14 pin socket.

NEXT

1. Insert the 40 pin socket into the socket at U 72, keeping pin 13 bent out.

2. Insert the 14 pin socket into the socket at U 73, keeping pins 5 & 6 bent out.

3. Solder the free end of the jumper attached to pin 6 of the 14 pin socket, to the base of pin 32 of J 6, the floppy drive cable. OPTION 1 Instead of this jumper you could use a four inch piece of test lead with a microclip on it and just clip it to J 6. Such test leads are available at Radio Shack for $3.95 a pair. They are small enough to avoid interfering with the floppy drive cable connector.

OPTION 2 For those of you brave (or experienced) enough, the jumper could be installed on the underside of the motherboard direct from pin 6 of U 73 to pin 32 of J 6.

4. Insert all the ICs back into their respective sockets with the exception of U 47, where you'll need a new EPROM, an U 73. The 74LS04 that was there is too slow, so it must be replaced with a 74S04, which is faster and more powerful.

For U 47, you'll need either an 81-232 EPROM (stock K-4) or one of the after market EPROMs, ADVENT, Micro Cornucopia, or what have you.

There is one more stage to this project, replacing the floppy drives, and since the motherboard is already out, this is a good time to do it. Double-sided double-density drives are also known in the

IBM PC/ Clone world as 360k drives and should be readily available locally at about $55 each NEW. There are also usually several good ones at any given swap meet. A HINT ... TEAC drives are some of the best around, very reliable, almost bullet-proof and in fact my favorites, BUT the card edge connector on the back of these beauties is exactly reversed from that on the original TANDONs, so keep an eye on the red stripe and the slot. If you look, you will be able to find a couple of full-height drives, and then replacement is just a matter of 4 1/8th inch allen screws (6-32 by 5/16ths), two power connectors, a couple of ground clips and the ribbon cable. If half-height drives are all that are available, they'll do just fine. In fact they will use less power than the full height stock ones. You will have to drill some extra mounting holes in the drive cage, but measure carefully, use a center punch, and you'll be all right. Fortunately the drive cage comes out easily if you remove the four screws from the bottom of the case.

THE NECESSARY STUFF LIST

Parts	**Sockets**
1	14 pin dip
1	16 pin dip
1	20 pin dip
1	24 pin dip
1	40 pin dip
	ICs
1	Kaypro IV EPROM (2732)
1	74S04
	Drives
2	Double-sided Double-density
	Tools
1	small screwdriver
1	#2 Phillips screwdriver (to remove the case and motherboard)
1	SMALL (15 watt) soldering iron
	miscellaneous wire, solder, flux

BITES .....

Benefits, Ideas, Techniques, Experiences & Serendipities

Several people have mentioned "screen shrinkage" when the drives start rotating. It is caused by a momentary fluctuation in the unregulated 12v coming from the power supply. I fixed mine by installing a larger power supply, but Lee Hart wrote and described a really unique rather elegant solution. He procured a 12v dc 2 amp plug-in transformer power supply like those that come with rechargeable calculators or small tape recorders. He then cut the 12v and ground wires going from the real power supply to the video board (under the CRT) and spliced them to the wires from the new transformer supply. He then cut the socket end and about a foot of wire from an extension cord, plugged in the transformer and connected the wires to the switch on the back of the Kaypro. The video now has its own power supply, and no more "screen shrinkage."

THE MARCH OF TIME

Those who read and memorize every word have undoubtedly realized that not only is this Issue # 60, but since there are 6 Issues a year, it is also TCJ's 10th anniversary. Coincidentally, it is also 10 years since the "Classic" CP/M computers became widely available, the prime examples of which are the K-II and the K-IV, not to mention a plethora of others such as Osbourne, Epson QX-10, Otrona, Morrow, etc.

At that time, since "we" were the only ones brave (stupid) enough to jump in with both feet, we were, as George Morrow said in "Thoughts of Chairman George", the "cognicenti", the"literati" of society. We did our job so well, preaching the gospel, that we worked ourselves right into Oblivion. Now everybody who is anybody, or who fancies themselves to be somebody, has more computer than they'll ever need, and is reasonably facile with it and their software.

Old habits die hard, however, and there will always be curiousbeginners, and that's where you'll find us and our "Classic" computers. This btw, is being written on my trusty 83 K-IV, bought new, (and they'll have to pry it from my cold dead hands), and now equipped with 2-800k floppies, 1-400k floppy, a 20 meg harddrive, an external composite monitor adapter (for software demos to clubs), 100 watt power supply, fan, and internal modem, all running at 5 Mhz.

NEXT TIME ...

The promised, and long awaited "speed-up". So...Keep those cards and letters coming.

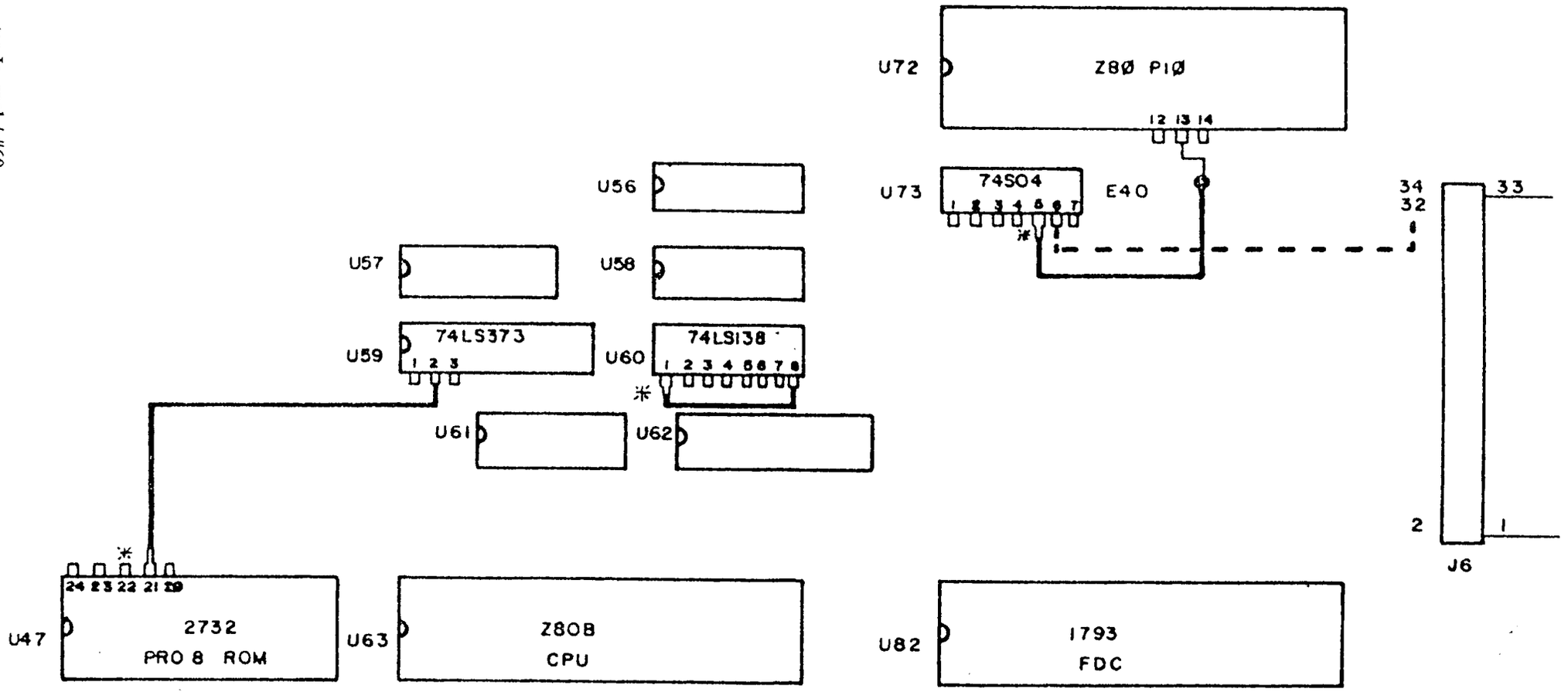
Thanks for the article Chuck, and BTW means "by the way." Chuck forgot to mention that he sells the upgrade ROMs if you can't find a local source. Send your request for help (or to order ROMs) to: Chuck Stafford, 4000 Norris Ave., Sacramento, CA 95821. BDK.

Advertising Rates For *The Computer Journal*

Size	1 Insertion	2-3 Insertions	4+Insertions
Full	**$400**	**$360**	**$320**
1/2 Page	**$240**	**$215**	**$195**
1/3 Page	**$195**	**$160**	**$145**
1/4 Page	**$160**	**$120**	**$100**

SPECIAL ANNIVERSARY ADVERTISING RATES

For the NEXT THREE ISSUES TCJ Advertising Rates will fall back to 1982/3 AMOUNTS!
That is HALF the ABOVE PRICES or get TWO ISSUES for the price of ONE.



Kaypro II to IV Modifications

Special Feature

Intermediate Users

Forth Kernel Design

MOVING FORTH

by Brad Rodriguez

Part 2: Benchmarks and Case Studies of Forth Kernels

In the last issue of TCJ, Brad covered the theoretical aspect of kernel design. Considerations were presented on variations due to different CPU selections. This time Brad shows actual examples of several CPU implementations. BDK.

BENCHMARKS

By now it must seem that the answer to every design question I presented in Part 1 is "code it and see." Obviously you don't want to write the entire Forth kernel several different ways just to evaluate different schemes. Fortunately, you can get quite a good "feel" with just a small subset of the Forth kernel.

Guy Kelly [KEL92] examines the following code samples for 19 different IBM PC Forths:

NEXT ...the "inner interpreter" that chains from one Forth word to another in the "thread". This is used at the end of every CODE definition, and is one of the most important factors in speed of Forth execution. You've already seen the pseudo-code for this in ITC and DTC; in STC it's just CALL/RETURN.

ENTER ...also called DOCOL or DOCOLON; the Code Field action that causes a high level "colon" definition to be executed. This, too, is crucial for speed; it is used at the start of every colon definition. Not needed in STC.

EXIT ...called ;S in fig-Forth; the code that ends the execution of a colon definition. This is essentially the high-level subroutine return, and appears at the end of every colon definition. This is just a machine code RETURN in STC.

NEXT, ENTER, and EXIT indicate the performance of the threading mechanism. These should be coded to evaluate ITC vs. DTC vs. STC. They also reflect the quality of your register assignments for IP, W, and RSP.

DOVAR ...a.k.a. "variable"; the machine code fragment that is the Code Field action for all Forth VARIABLEs.

DOCON ...a.k.a. "constant"; the machine code fragment that is the Code Field action for all Forth CONSTANTs.

DOCON and DOVAR, along with ENTER, show how efficiently you can obtain the Parameter Field address of a word being executed. This reflects your choice for the W register. In a DTC Forth, this also indicates whether to put a JUMP or CALL in the Code Field.

LIT ...a.k.a. "literal"; is a Forth word that fetches a cell value from the high-level thread. Several words use such in-line parameters, and this is a good indicator of their performance. It reflects your choice for the IP register.

@ ...the Forth memory-fetch operator, shows how quickly memory can be accessed from high-level Forth. This word usually benefits from TOS in stack.

! ...the Forth memory-store operator, is another indicator of memory access. This consumes two items from the stack, and illustrates efficiency of Parameter Stack access. It's a good indicator of the TOS-in-memory vs. TOS-in-register tradeoff.

+ ...the addition operator, is a representative example of all the Forth arithmetic and logical operators. Like the ! word, this benchmarks stack access, and it's a clear demonstration of any TOS-in-register benefit.

This is an excellent set of code samples. I have a few additional favorites:

DODOES ...is the Code Field action for words built with DOES>. This doesn't yield any new benchmark comparisons, although it does reflect the usefulness of W, IP, and RSP. I include it because it's the most convoluted code in the Forth kernel. If you can code the logic of DODOES, everything else is a snap. The intricacies of DODOES will be described in a subsequent article.

SWAP ...a simple stack operator, but still educational.

OVER ...a more complex stack operator. This gives a good idea of how easily you can access the Parameter Stack.

ROT ...a still more complex stack operator, and the one most likely to need an extra temporary register. If you can code ROT without needing an "X" register, you probably don't need an "X" register for anything.

0= ...one of the few unary arithmetic operators, and one of the most likely to benefit from TOS-in-register.

+! ...a most illustrative operator, combining stack access, arithmetic, memory fetch and store. This is one of my favorite benchmarks, although it is less frequently used than the other words in this list.

These are among the most-used words in the Forth kernel. It pays to optimize them. I'll show examples of all of these, including pseudo-code, for the 6809. For the other CPUs, I'll use selected examples to illustrate specific decisions.

CASE STUDY 1: THE 6809

In the world of 8-bit CPUs, the 6809 is the Forth programmer's dream machine. It supports two stacks! It also has two other address registers, and a wealth of orthogonal addressing modes second only to the PDP-11. ("Orthogonal" means they work the same way and have the same options for all address registers.) The two 8-bit accumulators can be treated as a single 16-bit accumulator, and there are many 16-bit operations.

The programmer's model of the 6809 is [MOT83]:

A - 8 bit accumulator
B - 8 bit accumulator

Most arithmetic operations use an accumulator as the destination. These can be concatenated and treated as a single 16-bit accumulator D (A high byte, B low).

X - 16 bit index register
Y - 16 bit index register
S - 16 bit stack pointer
U - 16 bit stack pointer

All addressing modes for X and Y can also be used with the S and U registers.

PC - 16 bit program counter
CC - 8 bit Condition Code register
DP - 8 bit Direct Page register

The 6800 family's Direct addressing mode uses an 8-bit address to reach any location in memory page zero. The 6809 allows any page to be Direct-addressed; this register provides the high 8 bits of address.

Those two stack pointers are crying out for Forth use. They are equivalent, except that S is used for subroutine calls and interrupts. Let's be consistent and use S for return addresses, leaving U for the Parameter Stack.

W and IP both need to be address registers, so these are the logical use for X and Y. X and Y are equivalent, so let's arbitrarily assign X=W, and Y=IP.

Now a threading model can be chosen. I'll scratch STC and TTC, to make this a "conventional" Forth. The limiting factor in performance is then the NEXT routine. Let's look at this in both ITC and DTC:

```
ITC-NEXT: LDX ,Y++          (8) (IP)->W, increment IP
          JMP [,X]          (6) (W)->temp, jump to adrs in temp

DTC-NEXT: JMP [,Y++]        (9) (IP)->temp, increment IP,
                                jump to adrs in temp

                            ("temp" is internal to the 6809)
```

NEXT is one instruction in a DTC 6809! This means you can code it in-line in two bytes, making it both smaller and faster than JMP NEXT. For comparison, look at the "NEXT" logic for subroutine threading:

```
          RTS               (5) ...at the end of one CODE word
          JSR nextword      (8) ...in the "thread"
          ...                   ...start of the next CODE word
```

STC takes 13 clocks to thread to the next word, compared with 9 clocks for DTC. This is because subroutine threading has to pop and push a return address, while simple DTC or ITC threading between CODE words does not.

Given the choice of DTC, you have to decide: does a high-level word have a Jump or Call in its Code Field? The driving consideration is how quickly can you obtain the address of the parameter field which follows? Let's look at the code to ENTER a colon definition, using symbolic Forth register names, to see this illustrated:

```
using a JSR (Call):
          JSR ENTER         (8)
          ...
ENTER:    PULS W            (7) get address following JSR into W reg
          PSHS IP           (7) save the old IP on the Return Stack
          TFR  W,IP         (6) Parameter Field address -> IP
          NEXT              (9) assembler macro for JMP [,Y++]
                            37 cycles total

using a JMP:
          JMP ENTER         (4)
          ...
ENTER:    PSHS IP           (7) save the old IP on the Return Stack
          LDX -2,IP         (6) re-fetch the Code Field address
          LEAY 3,X          (5) add 3 and put into IP (Y) register
          NEXT              (9)
                            31 cycles total

                            (CPU cycle counts are in parentheses.)
```

The DTC 6809 NEXT doesn't use the W register, because the 6809 addressing modes allow an extra level of indirection automatically. The JMP version of ENTER has to re-fetch the Code Field address -- NEXT didn't leave it in any register -- and then add 3 to get the Parameter Field address. The JSR version can get the Parameter Field address directly by popping the return stack. Even so, the JMP version is faster. (Exercise for the student: try coding the JSR ENTER with S=PSP and U=RSP.)

Either way, the code for EXIT is the same:

```
EXIT:     PULS IP           pop "saved" IP from return stack
          NEXT              continue Forth interpretation
```

Some registers remain to allocate. You could keep the User Pointer in memory, and this Forth would still be pretty fast. But the DP register would go to waste, and there's not much else it can do. Let's use the "trick" described above, and hold the high byte of UP in the DP register. (The low byte of UP is implied to be zero).

One 16-bit register is left: D. Most arithmetic operations need this register. Should it be left free as a scratch register, or used as the Top-Of-Stack? 6809 instructions use memory as one operand, so a second working register may be unnecessary. And if a scratch register is needed, it's easy to push and pop D. Let's write the benchmark primitives both ways, and see which is faster.

NEXT, ENTER, and EXIT don't use the stack, and thus have identical code either way.

DOVAR, DOCON, LIT, and OVER require the same number of CPU cycles either way. These illustrate the earlier comment that putting TOS in register often just changes where the push or pop takes place:

```
        TOS in D          TOS in memory..    .pseudo-code

DOVAR:  PSHU TOS          LDD  -2,IP         address of CF -> D
        LDD  -2,IP        ADDD #3            address of PF -> D
        ADDD #3           PSHU D             push D onto stack
        NEXT              NEXT

DOCON:  PSHU TOS          LDX  -2,IP         address of CF -> W
        LDX  -2,IP        LDD  3,X           contents of PF -> D
        LDD  3,X          PSHU D             push D onto stack
        NEXT              NEXT

LIT:    PSHU TOS          LDD  ,IP++         (IP) -> D, increment
IP
        LDD  ,IP++        PSHU D             push D onto stack
        NEXT              NEXT

OVER:   PSHU D            LDD  2,PSP         2nd on stack -> D
        LDD  2,PSP        PSHU D             push D onto stack
        NEXT              NEXT
```

SWAP, ROT, 0=, @, and especially + are all faster with TOS in register:

```
        TOS in D  TOS in memory...pseudo-code

SWAP:   LDX ,PSP (5)      LDD ,PSP (5)       TOS -> D
        STD ,PSP (5)      LDX 2,PSP (6)      2nd on stack -> X
        TFR X,D (6)       STD 2,PSP (6)      D -> 2nd on stack
        NEXT              STX ,PSP (5)       X -> TOS
                          NEXT

ROT:    LDX ,PSP (5)      LDX ,PSP (5)       TOS -> X
        STD ,PSP (5)      LDD 2,PSP (6)      2nd on stack -> D
        LDD 2,PSP (6)     STX 2,PSP (6)      X -> 2nd on stack
        STX 2,PSP (6)     LDX 4,PSP (6)      3rd on stack -> X
        NEXT              STD 4,PSP (6)      D -> 3rd on stack
                          STX ,PSP (5)       X -> TOS
                          NEXT

0=:     CMPD #0           LDD ,PSP           TOS -> D
        BEQ TRUE          CMPD #0            does D equal zero?
FALSE:LDD #0              BEQ TRUE
        NEXT              FALSE:LDD #0       no...put 0 in TOS
TRUE: LDD #-1             STD ,PSP
        NEXT              NEXT
                          TRUE: LDD #-1      yes...put -1 in TOS
                          STD ,PSP
                          NEXT

@:      TFR TOS,W (6)     LDD [,PSP] (8)     fetch D using TOS adrs
        LDD ,W (5)        STD ,PSP (5)       D -> TOS
        NEXT              NEXT

+:      ADDD ,U++         PULU D             pop TOS into D
        NEXT              ADDD ,PSP          add new TOS into D
                          STD ,PSP           store D into TOS
                          NEXT
```

! and +! are slower with TOS in register:

```
        TOS in D  TOS in memory...pseudo-code

!:      TFR TOS,W (6)     PULU W (7)         pop adrs into W
        PULU D (7)        PULU D (7)         pop data into D
        STD ,W (5)        STD ,W (5)         store data to adrs
        PULU TOS (7)      NEXT
        NEXT

+!:     TFR TOS,W (6)     PULU W (7)         pop adrs into W
        PULU TOS (7)      PULU D (7)         pop data into D
        ADDD ,W (6)       ADDD ,W (6)        add memory into D
        STD ,W (5)        STD ,W (5)         store D to memory
        PULU TOS (7)      NEXT
        NEXT
```

The reason these words are slower is that most Forth memory-reference words expect the address on the top of stack, so an extra TFR instruction is needed. This is why it's a help for the TOS register to be an address register. Unfortunately, all the 6809 address registers are spoken for...and it's much more important for W, IP, PSP, and RSP to be in address registers than TOS. The TOS-in-register penalty for ! and +! should be outweighed by the gains in the many arithmetic and stack operations.

CASE STUDY 2: THE 8051

If the 6809 is the Forthwright's dream machine, the 8051 is the nightmare. It has only one general-purpose address register, and one addressing mode, which always uses the one 8-bit accumulator.
All of the arithmetic operations, and many of the logical, must use the accumulator. The only 16-bit operation is INC DPTR. The hardware stack must use the 128-byte on-chip register file. [SIG92] Such a CPU could give ulcers.

Some 8051 Forths have been written that implement a full 16-bit model, e.g. [PAY90], but they are too slow for my taste. Let's make some tradeoffs and make a faster 8051 Forth.

Our foremost reality is the availability of only one address register. So let's use the 8051's Program Counter as IP -- i.e., let's make a subroutine-threaded Forth. If the compiler uses 2-

byte ACALLs instead of 3-byte LCALLs whenever possible, most of the STC code will be as small as ITC or DTC code.

Subroutine threading implies that the Return Stack Pointer is the hardware stack pointer. There are 64 cells of space in the on-chip register file, not enough room for multiple task stacks. At this point you can

a) restrict this Forth to single-task;

b) code all of the Forth definitions so that upon entry they move their return address to a software stack in external RAM; or

c) do task switches by swapping the entire Return Stack to and from external RAM.

Option (b) is slow! Moving 128 bytes on every task switch is faster than moving 2 bytes on every Forth word. For now I choose option (a), leaving the door open for (c) at some future date.

The one-and-only "real" address register, DPTR, will have to do multiple duty. It becomes W, the multi-purpose working register.

In truth, there are two other registers that can address external memory: R0 and R1. They provide only an 8-bit address; the high 8 bits are explicitly output on port 2. But this is a tolerable restriction for stacks, since they can be limited to a 256-byte space. So let's use R0 as the PSP.

This same 256-byte space can be used for user data. This makes P2 (port 2) the high byte of the User Pointer, and, like the 6809, the low byte will be implied to be zero.

What is the programmer's model of the 8051 so far?

reg adrs	8051 name	Forth usage
0	R0	low byte of PSP
1	R1	
2	R2	
3	R3	
4	R4	
5	R5	
6	R6	
7	R7	
8-7Fh		120 bytes of return stack
81h	SP	low byte of RSP (high byte=00)
82-83h	DPTR	W register
A0h	P2	high byte of UP and PSP
E0h	A	
F0h	B	

Note that this uses only register bank 0. The additional three register banks from 08h to 1Fh, and the bit-addressable region from 20h to 2Fh, are of no use to Forth. Using bank 0 leaves the largest contiguous space for the return stack. Later the return stack can be shrunk, if desired.

The NEXT, ENTER, and EXIT routines aren't needed in a subroutine threaded Forth.

What about the top of stack? There are plenty of registers, and memory operations on the 8051 are expensive. Let's put TOS in R3:R2 (with R3 as the high byte, in Intel fashion). Note that B:A can't be used -- the A register is the funnel through which all memory references must move!

Harvard architectures

The 8051 uses a "Harvard" architecture: program and data are kept in separate memories. (The Z8 and TMS320 are two other examples.) The 8051 is a degenerate case: there is physically no means to write to the program memory! This means that a Forthwright can do one of two things:

a) cross-compile everything, including the application, and give up all hope of putting an interactive Forth compiler on the 8051; or

b) cause some or all of the program memory to also appear in the data space. The easiest way is to make the two spaces completely overlap, by logically ORing the active-low PSEN* and RD* strobes with an external AND gate.

The Z8 and TMS320C25 are more civilized: they allow write access to program memory. The implications for the design of the Forth kernel will be discussed in subsequent articles.

CASE STUDY 3: THE Z80

The Z80 is instructive because it is an extreme example of a non-orthogonal CPU. It has four different kinds of address registers! Some operations use A as destination, some any 8-bit register, some HL, some any 16-bit register, and so on. Many operations (such as EX DE,HL) are only defined for one combination of registers.

In a CPU such as the Z80 (or 8086!), the assignment of Forth functions must be carefully matched to the capabilities of the CPU registers. Many more tradeoffs need to be evaluated, and often the only way is to write sample code for a number of different assignments. Rather than burden this article down endless permutations of Forth code, I'll present one register assignment based on many Z80 code experiments. It turns out that these choices can be rationalized in terms of the general principles outlined earlier.

I want a "conventional" Forth, although I will use direct threading. All of the "classical" virtual registers will be needed.

Ignoring the alternate register set, the Z80 has six address registers, with the following capabilities:

BC,DE - LD A indirect, INC, DEC
also exchange DE/HL

HL - LD r indirect, ALU indirect, INC, DEC, ADD, ADC, SBC, exchange w/TOS, JP indirect

IX,IY - LD r indexed, ALU indexed, INC, DEC, ADD, ADC, SBC, exchange w/TOS, JP indirect (all slow)

SP - PUSH/POP 16-bit, ADD/ADC/SUB to HL/IX/IY

BC, DE, and HL can also be manipulated in 8-bit pieces.

The 8-bit register A must be left as a scratch register, since it's the destination for so many ALU and memory reference operations.

HL is undoubtedly the most versatile register, and at one time or another it is tempting to use it for each of the Forth virtual registers. However, because of its versatility -- and because it is the only register which can be fetched byte-wise and used in an indirect jump -- HL should be used for W, Forth's all-purpose working register.

IX and IY might be considered for the Forth stack pointers, because of their indexed addressing mode, which can be used in ALU operations. But there are two problems with this: it leaves SP without a job; and, IX/IY are too slow! Most of the operations on either stack involve pushing or popping 16-bit quantities. This is one instruction using SP, but it requires four using IX or IY. One of the Forth stacks should use SP. And this should be the Parameter Stack, since it is used more heavily than the Return Stack.

What about Forth's IP? Mostly, IP fetches from memory and autoincrements, so there's no programming advantage to using IX/IY over BC/DE. But speed is of the essence with IP, and BC/DE are faster. Let's put IP in DE: it has the advantage of being able to swap with HL, which adds versatility.

A second Z80 register pair (other than W) will be needed for 16-bit arithmetic. Only BC is left, and it can be used for addressing or for ALU operations with A. But should BC be a second working register "X", or the top-of-stack? Only code will tell; for now, let's optimistically assume that BC=TOS.

This leaves the RSP and UP functions, and the IX and IY registers unused. IX and IY are equivalent, so let's assign IX=RSP, and IY=UP.

Thus the Z80 Forth register assignments are:

BC = TOS	IX = RSP
DE = IP	IY = UP
HL = W	SP = PSP

Now look at NEXT for the DTC Forth:

```
DTC-NEXT: LD A,(DE)     (7) (IP)->W, increment IP
          LD L,A        (4)
          INC DE        (6)
          LD A,(DE)     (7)
          LD H,A        (4)
          INC DE        (6)
          JP (HL)       (4) jump to address in W
```

alternate version (same number of clock cycles)

```
DTC-NEXT: EX DE,HL      (4) (IP)->W, increment IP
NEXT-HL:  LD E,(HL)     (7)
          INC HL        (6)
          LD D,(HL)     (7)
          INC HL        (6)
          EX DE,HL      (4)
          JP (HL)       (4) jump to address in W
```

Note that cells are stored low-byte first in memory. Also, although it might seem advantageous to keep IP in HL, it really isn't. This is because the Z80 can't JP (DE). The NEXT-HL entry point will be used shortly.

Just for comparison, let's look at an ITC NEXT. The pseudo-code given previously requires another temporary register "X", whose contents can be used for an indirect jump. Let DE=X, and BC=IP. TOS will have to be kept in memory.

```
ITC-NEXT: LD A,(BC)     (7) (IP)->W, increment IP
          LD L,A        (4)
          INC BC        (6)
          LD A,(BC)     (7)
          LD H,A        (4)
          INC BC        (6)

          LD E,(HL)     (7) (W)->X
          INC HL        (6)
          LD D,(HL)     (7)
          EX DE,HL      (4) jump to address in X
          JP (HL)       (4)
```

This leaves "W" incremented by one, and in the DE register. As long as this is done consistently, there's no problem -- code needing the contents of W knows where to find it, and how much to adjust it.

The ITC NEXT is 11 instructions, as compared to 7 for DTC. And ITC on the Z80 loses the ability to keep TOS in a register. My choice is DTC.
If coded in-line, DTC NEXT would require seven bytes in every CODE word. A jump to a common NEXT routine would only use three bytes, but would add 10 clock cycles. This is another of the tradeoff decisions in designing a Forth kernel. This example is a close call; let's opt for speed with an in-line NEXT. But sometimes NEXT is so huge, or memory is so tight, that the prudent decision is to use a JMP NEXT.

Now let's look at the code for ENTER. Using a CALL, the hardware stack is popped to get the Parameter Field address:

```
          CALL ENTER    (17)
          ...
ENTER:    DEC IX        (10) push the old IP on the return stack
```

```
        LD (IX+0),D     (19)
        DEC IX          (10)
        LD (IX+0),E     (19)
        POP DE          (10) Parameter Field address -> IP
        NEXT            (38) assembler macro for 7 instructions
```

Actually it's faster to POP HL, and then use the last six instructions of NEXT (omitting the EX DE,HL):

```
        CALL ENTER      (17)
        ...
ENTER:  DEC IX          (10) push the old IP on the return stack
        LD (IX+0),D     (19)
        DEC IX          (10)
        LD (IX+0),E     (19)
        POP HL          (10) Parameter Field address -> HL
        NEXT-HL         (34) see DTC NEXT code, above
                        119 cycles total
```

When a JP is used, the W register (HL) is left pointing to the Code Field. The Parameter Field is 3 bytes after:

```
        JP ENTER(10)
        ...
ENTER:  DEC IX          (10) push the old IP on the return stack
        LD (IX+0),D     (19)
        DEC IX          (10)
        LD (IX+0),E     (19)
        INC HL          ( 6) Parameter Field address -> IP
        INC HL          ( 6)
        INC HL          ( 6)
        NEXT-HL         (34)
                        120 cycles total
```

Again, because of the alternate entry point for NEXT, the new value for IP doesn't actually have to be put into the DE register pair.

The CALL version is one cycle faster. On an embedded Z80, a one-byte RST instruction could be used to gain speed and save space. This option is not available on many Z80-based personal computers.

CASE STUDY 4: THE 8086

The 8086 is another instructive CPU. Rather than go through the design process, let's look at one of the newer shareware Forths for the IBM PC: Pygmy Forth [SER90].

Pygmy is a direct-threaded Forth with the top-of-stack kept in register. The 8086 register assignments are:

AX = W	DI = scratch
BX = TOS	SI = IP
CX = scratch	BP = RSP
DX = scratch	SP = PSP

Most 8086 Forths use the SI register for IP, so that NEXT can be written with the LODSW instruction. In Pygmy the DTC NEXT is:

```
NEXT:   LODSW
        JMP AX
```

This is short enough to include in-line in every CODE word.

High-level and "defined" Forth words use a JMP (relative) to their machine code. The ENTER routine (called 'docol' in Pygmy) must therefore get the Parameter Field address from W:

```
ENTER:  XCHG SP,BP
        PUSH SI
        XCHG SP,BP
        ADD AX,3        Parameter Field address -> IP
        MOV SI,AX
        NEXT
```

Note the use of XCHG to swap the two stack pointers. This allows the use of PUSH and POP instructions for both stacks, which is faster than using indirect access on BP.

```
EXIT:   XCHG SP,BP
        POP SI
        XCHG SP,BP
        NEXT
```

Segment model

Pygmy Forth is a single-segment Forth; all code and data are contained within a single 64 Kbyte segment. (This is the "tiny model" in Turbo C lingo.) All of the Forth standards issued to date assume that everything is contained in a single memory space, accessible with the same fetch and store operators.

Nevertheless, IBM PC Forths are beginning to appear that use multiple segments for up to five different kinds of data [KEL92,SEY89]. These are:

CODE ...machine code
LIST ...high-level Forth threads (a.k.a. THREADS)
HEAD ...headers of all Forth words
STACK ...parameter and return stacks
DATA ...variables and user-defined data

This allows PC Forths to break the 64K limit, without going to the expense of implementing a 32-bit Forth on a 16-bit CPU. Implementation of a multi-segment model, and the ramifications for the Forth kernel, are beyond the scope of this article.

STILL TO COME...

Subsequent articles will look at:
- design tradeoffs in the Forth header and dictionary search
- the logic of CONSTANTs, VARIABLEs, and other data structures
- the defining word mechanisms, CREATE...;CODE and CREATE...DOES>
- the assembler vs. metacompiler question- the assembler and high-level code that comprises a Forth kernel
- multitasking modifications to the kernel

REFERENCES

[CUR93a] Curley, Charles, "Life in the FastForth Lane," awaiting publication in Forth Dimensions. Description of a 68000 subroutine-threaded Forth.

[CUR93b] Curley, Charles, "Optimizing in a BSR/JSR Threaded Forth," awaiting publication in Forth Dimensions. Single-pass code optimization for FastForth, in only five screens of code! Includes listing.

[KEL92] Kelly, Guy M., "Forth Systems Comparisons," Forth Dimensions XIII:6 (Mar/Apr 1992). Also published in the 1991 FORML Conference Proceedings. Both available from the Forth Interest Group, P.O. Box 2154, Oakland, CA 94621. Illustrates design tradeoffs of many 8086 Forths with code fragments and benchmarks -- highly recommended!

[KOG82] Kogge, Peter M., "An Architectural Trail to Threaded-Code Systems," IEEE Computer, vol. 15 no. 3 (Mar 1982). Remains the definitive description of various threading techniques.

[MOT83] Motorola Inc., 8-Bit Microprocessor and Peripheral Data, Motorola data book (1983).

[ROD91] Rodriguez, B.J., "B.Y.O. Assembler," Part 1, The Computer Journal #52 (Sep/Oct 1991). General principles of writing Forth assemblers.

[ROD92] Rodriguez, B.J., "B.Y.O. Assembler," Part 2, The Computer Journal #54 (Jan/Feb 1992). A 6809 assembler in Forth.

[SCO89] Scott, Andrew, "An Extensible Optimizer for Compiling Forth," 1989 FORML Conference Proceedings, Forth Interest Group, P.O. Box 2154, Oakland, CA 94621. Good description of a 68000 optimizer; no code provided.

[SIG92] Signetics Inc., 80C51-Based 8-Bit Microcontrollers, Signetics data book (1992).

Forth Implementations

[CUR86] Curley, Charles, real-Forth for the 68000, privately distributed (1986).

[JAM80] James, John S., fig-Forth for the PDP-11, Forth Interest Group (1980).

[KUN81] Kuntze, Robert E., MVP-Forth for the Apple II, Mountain View Press (1981).

[LAX84] Laxen, H. and Perry, M., F83 for the IBM PC, version 2.1.0 (1984). Distributed by the authors, available from the Forth Interest Group or GEnie.

[LOE81] Loeliger, R. G., Threaded Interpretive Languages, BYTE Publications (1981), ISBN 0-07-038360-X. May be the only book ever written on the subject of creating a Forth-like kernel (the example used is the Z80). Worth it if you can find a copy.

[MPE92] MicroProcessor Engineering Ltd., MPE Z8/Super8 PowerForth Target, MPE Ltd., 133 Hill Lane, Shirley, Southampton, S01 5AF, U.K. (June 1992). A commercial product.

[PAY90] Payne, William H., Embedded Controller FORTH for the 8051 Family, Academic Press (1990), ISBN 0-12-547570-5. This is a complete "kit" for a 8051 Forth, including a metacompiler for the IBM PC. Hardcopy only; files can be downloaded from GEnie. Not for the novice!

[SER90] Sergeant, Frank, Pygmy Forth for the IBM PC, version 1.3 (1990). Distributed by the author, available from the Forth Interest Group. Version 1.4 is now available on GEnie, and worth the extra effort to obtain.

[SEY89] Seyward, H., Elehew, W. R., and Caven, P., LOVE-83Forth for the IBM PC, version 1.20 (1989). A shareware Forth using a five-segment model. Contact Seyward Associates, 265 Scarboro Cres., Scarborough, Ontario M1M 2J7 Canada.

[TAL80] Talbot, R. J., fig-Forth for the 6809, Forth Interest Group (1980).

AUTHOR'S BIOGRAPHY

After twelve years of designing and programming embedded systems, Brad Rodriguez decided he didn't know everything, and went back to school. He is now working full time toward a Ph.D. in Computer Engineering, focusing on real-time applications of artificial intelligence. He still does a little work "on the side" as T-Recursive Technology, and can be contacted as bradford@maccs.dcss.mcmaster.ca on the Internet, or more promptly as B.RODRIGUEZ2 on GEnie. The telecommunicationally disadvantaged can write to him at Box 77, McMaster University, 1280 Main St. West, Hamilton, Ontario L8S 1C0 Canada.

COMING IN ISSUE #61

Multiprocessing For The Impoverished
by Brad Rodriguez

Part 1: a 6809 Uniprocessor

Brad builds a 6809 single board computer using junk box parts. Theory and Schematics provided for your hardware pleasure.

K-42
PASCAL RUNOFF—GRAPHICS

42-DISK	DOC	2k	D3C	IQC	3k	DRAW	IQC	12k
CRC	COM	3k	DD-DEMO	SQR	1k	DRAW	PQS	11k
CURSOR	COM	9k	DDRAW-20	COM	26k	ORBIT	COM	17k
CURSOR	PQS	2k	DDRAW-20	PQS	5k	ORBIT	IQC	3k
D	COM	3k	DDRAW2	DQC	8k	ORBIT	PQS	5k
D-DEMO	DAT	3k	DDRSUBS2	PQS	16k	ORBITHLP	IQC	4k
D3A	IQC	8k	DRAW	COM	31k	USQ	COM	2k
D3B	IQC	10k	DRAW	DQC	4k			

For you 84 Kaypro owners here are the best of the graphics programs submitted in our Pascal Runoff.

DRAW and**DDRAW** Both of these programs take full advantage of the Kaypro's block graphics to edit, rotate, merge, etc. any object you draw on the screen.

ORBIT This was one of our favorites. Sit back and watch a comet or planet zip around the screen.

CURSOR Set block or underline, blink or no-blink, and fast or slow blink rate.

K-43
PASCAL RUNOFF—GAMES

43-DISK	DOC•	2k	GENERAL	TXT•	1k	PLAYING	INC•	2k
ALPHABET	IQX•	4k	GUESSUP	INC•	2k	QUITS	INC•	1k
BEEP	INC•	1k	HELP	INC•	1k	RANDOM	IQX•	5k
C4	COM•	12k	INSTRUCT	QQQ•	5k	SCREENUP	INC•	2k
C4	PQS•	5k	KAYPRO+	TXT•	1k	SETTEST	INC•	1k
CHANWOLI	IQC•	1k	LANDER	COM•	21k	SETUP	INC•	2k
CHOOSCRT	INC•	2k	LANDER	DOC•	2k	SWAPSETS	IQC•	2k
CLEANUP	INC•	1k	LANDER	IQC•	13k	TGAME	COM	13k
CLEARBOX	INC•	1k	LANDER	PQS•	3k	TGAME	PQS	8k
CRC	COM•	3k	LANDER	SCO•	1k	UPDATE	IQC•	2k
CRCKLIST	CRC•	2k	LANDER	TYP•	1k	USQ	COM•	2k
DICE	COM•	14k	MAP	COM•	24k	VIEWLIST	INC•	1k
DICE	PQS•	8k	MAP	PQS•	5k	WORDLIST	DTA•	1k
DOMENU	INC•	2k	MENU	QQQ•	1k	WORDLIST	QQQ•	2k
DRIVERS	INC•	1k	MODREC	•	2k			
EDITOR	INC•	2k	PLAY	QQQ•	1k			

Five games from our Pascal Runoff competition.

K-44
PASCAL RUNOFF—PRINTER UTILITIES

44-DISK	DOC	2k	ITALOKI	CHR	2k	ROMANOKI	CHR	2k
CRC	COM	3k	LX80	COM	12k	SYM1OKI	CHR	2k
CRCKLIST	CRC	2k	LX80	PQS	5k	SYM2OKI	CHR	2k
FANFOLD	COM	15k	OKIUP	ASM	4k	USQ	COM	2k
FANFOLD	DOC	2k	OKIUP	COM	1k	XPRINT	COM	10k
FANFOLD	PQS	13k	OKIUTL	DQC	19k	XPRINT	FNT	2k
FX100	COM	12k	OKIUTL12	COM	21k	XPRINT	PQS	5k
FX100	PQS	4k	OKIUTL12	PQS	5k	XPRINTDB	COM	12k
GOTHOKI	CHR	2k	OKU12I1	PQS	7k	XPRINTDB	PQS	4k
ITALICS	CHR	2k	OKU12I2	PQS	8k			

FANFOLD Print on both sides of the paper with this slick program.

FX100 and LX80 Configure anything that can be configured on these two Epson printers.

OKIUTL12 In addition to normal printer configuration, OKIUTL allows you to edit your own custom character set on Okidata 92 and 93 printers.

XPRINT Spreadsheets and wide documents look great printed sideways. Written for the Epson MX-80 with Graftrax.

XPRINTDB Create and edit fonts for XPRINT. One character set is included.

K-45
PASCAL RUNOFF—UTILITIES

45-DISK	DOC	2k	SORTLINE	COM	14k	TDS	INC	9k
CRC	COM	3k	SORTLINE	PAS	6k	TDSS	INC	14k
CRCKLIST	CRC	1k	SRT	DOC	7k	TDTRAP	INC	2k
CYFER	COM	12k	SRT	PAS	14k	TDU	INC	3k
CYFER	PQS	7k	TD	COM	14k	USQ	COM	2k
NFILELST	COM	14k	TD	DOC	23k			
NFILELST	DOC	15k	TD	PAS	6k			
NFILELST	PQS	10k	TDR	INC	5k			

We received a number of general utilities as entries in our Turbo Pascal Runoff. This disk contains four of the best.

K-46
PASCAL RUNOFF—TURBO UTILITIES

46-DISK	DOC	2k	INLINE	DOC	3k	TRUN2C	COM	10k
CLKDEMO	COM	11k	INLINE	PAS	6k	TRUN2C	PAS	5k
CLKDEMO	PAS	6k	KAYPRO84	BOX	2k	TRUN3	COM	10k
CLOCK84	BOX	6k	SCREEN	COM	17k	TRUN3	PAS	4k
CRC	COM	3k	SCREEN	DOC	4k	VIDDEMO	COM	9k
CRCKLIST	CRC	2k	SCREEN	PAS	19k	VIDDEMO	PAS	2k
GRAFIX	BOX	4k	SETCLOCK	COM	9k	VIDEO84	BOX	4k
GRFDEMO	COM	11k	TRUN	DOC	8k			
GRFDEMO	PAS	5k	TRUN28	COM	10k			
INLINE	COM	10k	TRUN28	PAS	5k			

84TOOLBOX This group of files demonstrates how to control the video and real time clock capabilities of the 84 series Kaypros.

INLINE creates an inline statement using the PRN file generated by either the Z80MR or ASM assembler.

SCREEN With SCREEN's editor you can fiddle with a menu until it's just what you want, then hit ESC and a procedure to print the menu is created.

TRUN allows execution of chain files (no run time library).

K-47
256K SOFTWARE

47-DISK	DOC	4k	KAY256	LBR	58k	RAMINIT	COM	1k
256KFIX	DOC	1k	MT256	COM	1k	RAMINIT	Z80	1k
CBIOSR	DOC	6k	R2	COM	2k	USQ	COM	2k
CBIOSR	HEX	2k	R4-10	COM	2k	Z3BIOS4G	DOC	2k
CBIOSR	Z80	13k	RAMDRIVE	COM	2k	Z3BIOS4G	Z80	24k
CRC	COM	3k	RAMDRIVE	DOC	1k			
CRCKLIST	CRC	1k	RAMDRIVE	MAC	16k			

These programs support the 256K RAM upgrades published in *Micro C* issues #30 (for 83 Kaypros) and #34 (for 84 Kaypros).

K-48
C CONTEST WINNERS—I

48-DISK	DOC	2k	CITIES	WRD	1k	MOVE	C	6k
APUZZLE		1k	CRC	COM	3k	MOVEGEN	CQ	13k
BKG	C	33k	CRCKLIST	CRC	1k	SCREEN	CQ	10k
BKG	COM	38k	EVAL	CQ	8k	USQ	COM	2k
BKG	DOC	32k	IBMUTILS	C	5k	WORDSRCH	C	16k
BKG	H	2k	KAYUTILS	C	4k	WORDSRCH	DOC	7k

Our C programming contest attracted a number of games. The two on this disk (Backgammon and Wordsearch) were among the winners. Wordsearch comes in source only—Q/C compiler required for execution.

K-49
C CONTEST WINNERS—II

49-DISK	DOC	2k	PP	COM	29k	PP6	CQ	4k
CRC	COM	3k	PP	DQC	25k	PP7	C	4k
CRCKLIST	CRC	1k	PP	HQ	11k	PP8	CQ	10k
CTYPE	C	1k	PP1	CQ	13k	PPBANNER	C	1k
CTYPE	H	1k	PP2	CQ	13k	SETCLK	COM	4k
LOG	COM	10k	PP3	CQ	13k	SETCLK	CQ	6k
LOG	CQ	13k	PP4	CQ	9k	USQ	COM	2k
LOG	DOC	7k	PP5	CQ	3k			

Two very useful CP/M utilities from our C programming contest reside on this disk—a C preprocessor and a file time and date stamper.

The Computer Journal

Back Issues

Sales limited to supplies in stock.

Issues 1 to 19 are currently OUT of print. To assist those who want a full collection of TCJ issues we are preparing photo-copied sets. The sets will be issue 1 to 9 and 10 to 19. Each set will be bound with a plastic protective cover.

The price for bound volumes is $20 plus shipping . Expect TWO to THREE weeks for delivery after payment received at TCJ. Some single copies available, contact TCJ before ordering.

Issue Number 20:

- Designing an 8035 SBC
- Using Apple Graphics from CP/M: Turbo Pascal Controls Apple Graphics
- Soldering & Other Strange Tales
- Build an S-100 Floppy Disk Controller: WD2797 Controller for CP/M 68K

Issue Number 21:

- Extending Turbo Pascal: Customize with Procedures & Functions
- Unsoldering: The Arcane Art
- Analog Data Acquisition & Control: Connecting Your Computer to the Real World
- Programming the 8035 SBC

Issue Number 22:

- NEW-DOS: Write Your Own Operating System
- Variability in the BDS C Standard Library
- The SCSI Interface: Introductory Column
- Using Turbo Pascal ISAM Files
- The Ampro Little Board Column

Issue Number 23:

- C Column: Flow Control & Program Structure
- The Z Column: Getting Started with Directories & User Areas
- The SCSI Interface: Introduction to SCSI
- NEW-DOS: The Console Command Processor
- Editing the CP/M Operating System
- INDEXER: Turbo Pascal Program to Create an Index
- The Ampro Little Board Column

Issue Number 24:

- Selecting & Building a System
- The SCSI Interface: SCSI Command Protocol
- Introduction to Assemble Code for CP/M
- The C Column: Software Text Filters
- Ampro 186 Column: Installing MS-DOS Software
- The Z-Column
- NEW-DOS: The CCP Internal Commands
- ZTime-1: A Real Time Clock for the Ampro Z-80 Little Board

Issue Number 26:

- Bus Systems: Selecting a System Bus
- Using the SB180 Real Time Clock
- The SCSI Interface: Software for the SCSI Adapter
- Inside Ampro Computers
- NEW-DOS: The CCP Commands (continued)
- ZSIG Corner
- Affordable C Compilers
- Concurrent Multitasking: A Review of DoubleDOS

Issue Number 27:

- 68000 TinyGiant: Hawthorne's Low Cost 16-bit SBC and Operating System
- The Art of Source Code Generation: Disassembling Z-80 Software
- Feedback Control System Analysis: Using Root Locus Analysis & Feedback Loop Compensation
- The C Column: A Graphics Primitive Package
- The Hitachi HD64180: New Life for 8-bit Systems
- ZSIG Corner: Command Line Generators and Aliases
- A Tutor Program in Forth: Writing a Forth Tutor in Forth
- Disk Parameters: Modifying the CP/M Disk Parameter Block for Foreign Disk Formats

Issue Number 28:

- Starting Your Own BBS
- Build an A/D Converter for the Ampro Little Board
- HD64180: Setting the Wait States & RAM Refresh using PRT & DMA
- Using SCSI for Real Time Control
- Open Letter to STD Bus Manufacturers
- Patching Turbo Pascal
- Choosing a Language for Machine Control

Issue Number 29:

- Better Software Filter Design
- MDISK: Adding a 1 Meg RAM Disk to Ampro Little Board, Part 1
- Using the Hitachi hd64180: Embedded Processor Design
- 68000: Why use a new OS and the 68000?
- Detecting the 8087 Math Chip
- Floppy Disk Track Structure
- The ZCPR3 Corner

Issue Number 30:

- Double Density Floppy Controller
- ZCPR3 IOP for the Ampro Little Board
- 3200 Hackers' Language
- MDISK: Adding a 1 Meg RAM Disk to Ampro Little Board, Part 2
- Non-Preemptive Multitasking
- Software Timers for the 68000
- Lilliput Z-Node
- The ZCPR3 Corner
- The CP/M Corner

Issue Number 31:

- Using SCSI for Generalized I/O
- Communicating with Floppy Disks: Disk Parameters & their variations
- XBIOS: A Replacement BIOS for the SB180
- K-OS ONE and the SAGE: Demystifying Operating Systems
- Remote: Designing a Remote System Program
- The ZCPR3 Corner: ARUNZ Documentation

Issue Number 32:

- Language Development: Automatic Generation of Parsers for Interactive Systems
- Designing Operating Systems: A ROM based OS for the Z81
- Advanced CP/M: Boosting Performance
- Systematic Elimination of MS-DOS Files: Part 1, Deleting Root Directories & an In-Depth Look at the FCB
- WordStar 4.0 on Generic MS-DOS Systems: Patching for ASCII Terminal Based Systems
- K-OS ONE and the SAGE: System Layout and Hardware Configuration
- The ZCPR3 Corner: NZCOM and ZCPR34

Issue Number 33:

- Data File Conversion: Writing a Filter to Convert Foreign File Formats
- Advanced CP/M: ZCPR3PLUS & How to Write Self Relocating Code
- DataBase: The First in a Series on Data Bases and Information Processing
- SCSI for the S-100 Bus: Another Example of SCSI's Versatility
- A Mouse on any Hardware: Implementing the Mouse on a Z80 System
- Systematic Elimination of MS-DOS Files: Part 2, Subdirectories & Extended DOS Services
- ZCPR3 Corner: ARUNZ Shells & Patching WordStar 4.0

Issue Number 34:

- Developing a File Encryption System.
- Database: A continuation of the data base primer series.
- A Simple Multitasking Executive: Designing an embedded controller multitasking executive.
- ZCPR3: Relocatable code, PRL files, ZCPR34, and Type 4 programs.
- New Microcontrollers Have Smarts: Chips with BASIC or Forth in ROM are easy to program.
- Advanced CP/M: Operating system extensions to BDOS and BIOS, RSXs for CP/M 2.2.
- Macintosh Data File Conversion in Turbo Pascal.
- The Computer Corner

Issue Number 35:

- All This & Modula-2: A Pascal-like alternative with scope and parameter passing.
- A Short Course in Source Code Generation: Disassembling 8088 software to produce modifiable assem. source code.
- Real Computing: The NS32032.
- S-100: EPROM Burner project for S-100 hardware hackers.
- Advanced CP/M: An up-to-date DOS, plus details on file structure and formats.
- REL-Style Assembly Language for CP/M and Z-System. Part 1: Selecting your assembler, linker and debugger.
- The Computer Corner

Issue Number 36:

- Information Engineering: Introduction.
- Modula-2: A list of reference books.
- Temperature Measurement & Control: Agricultural computer application.
- ZCPR3 Corner: Z-Nodes, Z-Plan, Amstrand computer, and ZFILE.
- Real Computing: NS32032 hardware for experimenter, CPUs in series, software options.
- SPRINT: A review.
- REL-Style Assembly Language for CP/M & ZSystems, part 2.
- Advanced CP/M: Environmental programming.
- The Computer Corner.

Issue Number 37:

- C Pointers, Arrays & Structures Made Easier: Part 1, Pointers.
- ZCPR3 Corner: Z-Nodes, patching for NZCOM, ZFILER.
- Information Engineering: Basic Concepts: fields, field definition, client worksheets.
- Shells: Using ZCPR3 named shell variables to store date variables.
- Resident Programs: A detailed look at TSRs & how they can lead to chaos.
- Advanced CP/M: Raw and cooked console I/O.
- Real Computing: The NS 32000.
- ZSDOS: Anatomy of an Operating System: Part 1.
- The Computer Corner.

Issue Number 38:

- C Math: Handling Dollars and Cents With C.
- Advanced CP/M: Batch Processing and a New ZEX.
- C Pointers, Arrays & Structures Made Easier: Part 2, Arrays.
- The Z-System Corner: Shells and ZEX, new Z-Node Central, system security under Z-Systems.
- Information Engineering: The portable Information Age.
- Computer Aided Publishing: Introduction to publishing and Desk Top Publishing.
- Shells: ZEX and hard disk backups.
- Real Computing: The National Semiconductor NS320XX.
- ZSDOS: Anatomy of an Operating System, Part 2.

Issue Number 39:

- Programming for Performance: Assembly Language techniques.
- Computer Aided Publishing: The Hewlett Packard LaserJet.
- The Z-System Corner: System enhancements with NZCOM.
- Generating LaserJet Fonts: A review of Digi-Fonts.
- Advanced CP/M: Making old programs Z-System aware.
- C Pointers, Arrays & Structures Made Easier: Part 3: Structures.
- Shells: Using ARUNZ alias with ZCAL.
- Real Computing: The National Semiconductor NS320XX.
- The Computer Corner.

Issue Number 40:

- Programming the LaserJet: Using the escape codes.
- Beginning Forth Column: Introduction.
- Advanced Forth Column: Variant Records and Modules.
- LINKPRL: Generating the bit maps for PRL files from a REL file.
- WordTech's dBXL: Writing your own custom designed business program.
- Advanced CP/M: ZEX 5.0×The machine and the language.
- Programming for Performance: Assembly language techniques.
- Programming Input/Output With C: Keyboard and screen functions.
- The Z-System Corner: Remote access systems and BDS C.
- Real Computing: The NS320XX
- The Computer Corner.

Issue Number 41:

- Forth Column: ADTs, Object Oriented Concepts.
- Improving the Ampro LB: Overcoming the 88Mb hard drive limit.
- How to add Data Structures in Forth
- Advanced CP/M: CP/M is hacker's haven, and Z-System Command Scheduler.
- The Z-System Corner: Extended Multiple Command Line, and aliases.
- Programming disk and printer functions with C.
- LINKPRL: Making RSXes easy.
- SCOPY: Copying a series of unrelated files.
- The Computer Corner.

Issue Number 42:

- Dynamic Memory Allocation: Allocating memory at runtime with examples in Forth.
- Using BYE with NZCOM.
- C and the MS-DOS Screen Character Attributes.

The Computer Journal Back Issues

· Forth Column: Lists and object oriented Forth.
· The Z-System Corner: Genie, BDS Z and Z-System Fundamentals.
· 68705 Embedded Controller Application: An example of a single-chip microcontroller application.
· Advanced CP/M: PluPerfect Writer and using BDS C with REL files.
· Real Computing: The NS 32000.
· The Computer Corner

Issue Number 43:

· Standardize Your Floppy Disk Drives.
· A New History Shell for ZSystem.
· Heath's HDOS, Then and Now.
· The ZSystem Corner: Software update service, and customizing NZCOM.
· Graphics Programming With C: Graphics routines for the IBM PC, and the Turbo C graphics library.
· Lazy Evaluation: End the evaluation as soon as the result is known.
· S-100: There's still life in the old bus.
· Advanced CP/M: Passing parameters, and complex error recovery.
· Real Computing: The NS32000.
· The Computer Corner.

Issue Number 44:

· Animation with Turbo C Part 1: The Basic Tools.
· Multitasking in Forth: New Micros F68FC11 and Max Forth.
· Mysteries of PC Floppy Disks Revealed: FM, MFM, and the twisted cable.
· DosDisk: MS-DOS disk format emulator for CP/M.
· Advanced CP/M: ZMATE and using lookup and dispatch for passing parameters.
· Real Computing: The NS32000.
· Forth Column: Handling Strings.
· Z-System Corner: MEX and telecommunications.
· The Computer Corner

Issue Number 45:

· Embedded Systems for the Tenderfoot: Getting started with the 8031.
· The Z-System Corner: Using scripts with MEX.
· The Z-System and Turbo Pascal: Patching TURBO.COM to access the Z-System.
· Embedded Applications: Designing a Z80 RS-232 communications gateway, part 1.
· Advanced CP/M: String searches and tuning Jetfind.
· Animation with Turbo C: Part 2, screen interactions.
· Real Computing: The NS32000.
· The Computer Corner.

Issue Number 46:

· Build a Long Distance Printer Driver.
Using the 8031's built-in UART for serial communications.
· Foundational Modules in Modula 2.
· The Z-System Corner: Patching The Word Plus spell checker, and the ZMATE macro text editor.
· Animation with Turbo C: Text in the graphics mode.
· Z80 Communications Gateway: Prototyping, Counter/Timers, and using the Z80 CTC.

Issue Number 47:

· Controlling Stepper Motors with the 68HC11F
· Z-System Corner: ZMATE Macro Language
· Using 8031 Interrupts
· T-1: What it is & Why You Need to Know
· ZCPR3 & Modula, Too
· Tips on Using LCDs: Interfacing to the 68HC705
· Real Computing: Debugging, NS32 Multitasking & Distributed Systems
· Long Distance Printer Driver: correction
· ROBO-SOG 90
· The Computer Corner

Issue Number 48:

· Fast Math Using Logarithms
· Forth and Forth Assembler
· Modula-2 and the TCAP
· Adding a Bernoulli Drive to a CP/M Computer (Building a SCSI Interface)
· Review of BDS "Z"
· PMATE/ZMATE Macros, Pt. 1
· Real Computing
· Z-System Corner: Patching MEX-Plus and TheWord, Using ZEX
· Z-Best Software
· The Computer Corner

Issue Number 49:

· Computer Network Power Protection
· Floppy Disk Alignment w/RTXEB, Pt. 1
· Motor Control with the F68HC11
· Controlling Home Heating & Lighting, Pt. 1
· Getting Started in Assembly Language
· LAN Basics
· PMATE/ZMATE Macros, Pt. 2
· Real Computing
· Z-System Corner
· Z-Best Software
· The Computer Corner

Issue Number 50:

· Offload a System CPU with the Z181
· Floppy Disk Alignment w/RTXEB, Pt. 2
· Motor Control with the F68HC11
· Modula-2 and the Command Line
· Controlling Home Heating & Lighting, Pt. 2
· Getting Started in Assembly Language Pt 2
· Local Area Networks
· Using the ZCPR3 IOP
· PMATE/ZMATE Macros, Pt. 3
· Z-System Corner, PCED
· Z-Best Software
· Real Computing, 32FX16, Caches
· The Computer Corner

Issue Number 51:

· Introducing the YASBEC
· Floppy Disk Alignment w/RTXEB, Pt 3
· High Speed Modems on Eight Bit Systems
· A Z8 Talker and Host
· Local Area Networks--Ethernet
· UNIX Connectivity on the Cheap
· PC Hard Disk Partition Table
· A Short Introduction to Forth
· Stepped Inference as a Technique for Intelligent Real-Time Embedded Control
· Real Computing, the 32CG160, Swordfish, DOS Command Processor
· PMATE/ZMATE Macros
· Z-System Corner, The Trenton Festival
· Z-Best Software, the Z3HELP System
· The Computer Corner

Issue Number 52:

· YASBEC, The Hardware
· An Arbitrary Waveform Generator, Pt. 1
· B.Y.O. Assembler...in Forth
· Getting Started in Assembly Language, Pt. 3
· The NZCOM IOP
· Servos and the F68HC11
· Z-System Corner, Programming for Compatibility
· Z-Best Software
· Real Computing, X10 Revisited
· PMATE/ZMATE Macros
· Controlling Home Heating & Lighting, Pt. 3
· The CPU280, A High Performance Single-Board Computer
· The Computer Corner

Issue Number 53:

· The CPU280
· Local Area Networks
· Am Arbitrary Waveform Generator
· Real Computing
· Zed Fest '91
· Z-System Corner
· Getting Started in Assembly Language
· The NZCOM IOP
· Z-BEST Software
· The Computer Corner

Issue Number 54:

· Z-System Corner
· B.Y.O. Assembler
· Local Area Networks
· Advanced CP/M
· ZCPR on a 16-Bit Intel Platform
· Real Computing
· Interrupts and the Z80
· 8 MHZ on a Ampro
· Hardware Heavenn
· What Zilog never told you about the Super8
· An Arbitary Waveform Generator
· The Development of TDOS
· The Computer Corner

Issue Number 55:

· Fuzzilogy 101
· The Cyclic Redundancy Check in Forth
· The Internetwork Protocol (IP)
· Z-System Corner
· Hardware Heaven
· Real Computing
· Remapping Disk Drives through the Virtual BIOS
· The Bumbling Mathmatician
· YASMEM
· Z-BEST Software
· The Computer Corner

Issue Number 56:

· TCJ - The Next Ten Years
· Input Expansion for 8031
· Connecting IDE Drives to 8-Bit Systems
· Real Computing
· 8 Queens in Forth
· Z-System Corner
· Kaypro-84 Direct File Transfers
· Analog Signal Generation
· The Computer Corner

Issue Number 57:

· Home Automation with X10
· File Transfer Protocols
· MDISK at 8 MHZ.
· Real Computing
· Shell Sort in Forth
· Z-System Corner
· Introduction to Forth
· DR. S-100
· Z AT Last!
· The Computer Corner

Issue Number 58:

· Multitasking Forth
· Computing Timer Values
· Affordable Development Tools
· Real Computing
· Z-System Corner
· Mr. Kaypro
· DR. S-100
· The Computer Corner

Issue Number 59:

· Moving Forth
· Center Fold IMSAI MPU-A
· Developing Forth Applications
· Real Computing
· Z-System Corner
· Mr. Kaypro Review
· DR. S-100
· The Computer Corner

SPECIAL DISCOUNT

15% on cost of Back Issues when buying from 1 to Current Issue.

10% on cost of Back Issues when buying 20 or more issues.

Maximum Cost for shipping is $25.00 for U.S.A. and $45.00 for all other Countries.

	U.S.	Canada/Mexico		Europe/Other	
Subscriptions (CA not taxable)		(Surface)	(Air)	(Surface)	(Air)
1year (6 issues)	$24.00	$32.00	$34.00	$34.00	$44.00
2 years (12 issues)	$44.00	$60.00	$64.00	$64.00	$84.00
Back Issues (CA tax)	add these shipping costs for each issue ordered				
Bound Volumes $20.00 ea	+$3.00	+$3.50	+$6.50	+$4.00	+$17.00
#20 thru #43 are $3.00 ea.	+$1.00	+$1.00	+$1.25	+$1.50	+$2.50
#44 and up are $4.00ea.	+$1.25	+$1.25	+$1.75	+$2.00	+$3.50
Software Disks (CA tax) add these shipping costs for groups of 3 disks ordered					
MicroC Disks are $6.00ea	+$1.00	+$1.00	+$1.25	+$1.50	+$2.50

Items: _____

_____

Back Issues Total _____

MicroC Disks Total _____

California state Residents add 7.25% Sales TAX _____

Subscription Total _____

Total Enclosed _____

Name: _____

Address: _____

_____

_____

_____

Credit Card # _____-_____-_____-_____ exp ___/___

Payment is accepted by check, money order, or Credit Card. Checks must be in US funds, drawn on a US bank. Credit Card orders can call 1(800) 424-8825.

TCJ *The Computer Journal*

P.O. Box 535, Lincoln, CA 95648-0535
Phone (916) 645-1670

The Computer Journal - Micro Cornucopia Kaypro Disks

K1
MODEM PROGRAMS

K2
CP/M UTILITIES

K3
GAMES

K4
ADVENTURE

K5
MX80/GEM 10X GRAPHICS

K6
TEXT UTILITIES

K7
SMALL C VER 2

K8
SOURCE OF SMALL C

K9
GENERAL UTILITIES

K10
Z80 AND LINKING ASSEM

K11
CHECKBOOK PROGRAM &

K12
KAYPRO FORTH

K13
SOURCE OF FIG-FORTH

K14
SMARTMODEM PROGRAMS

K15
HARD DISK UTILITIES

K16
PASCAL COMPILER

K17
Z80 TOOLS

K18
SYSTEM DIAGNOSTICS

K19
PROWRITER GRAPHICS

K20
MICROSHERE'S COLOR
GRAPHICS BOARD

K21
SBASIC & SCREEN DUMP

K22
ZCPR

K23
FAST TERMINAL &
RCPM UTILITIES

K24
KEYBOARD TRANSLATOR &
MBASIC GAMES

K25
Z80 MACRO ASSEMBLER

K26
EPROM PROGRAMMER/TOOLS

K27
TYPING TUTORIAL

K28
MODEM 730 SOURCE

K29
TURBO PASCAL GAMES I

K30
TURBO PASCAL GAMES II

K31
TURBO BULLETIN BOARD

K32
FORTH-83

K33
UTILITIES

K34
GAMES

K35
SMALL C VER 2.1

K36
SMALL C LIBRARY

K37
UTILITIES PRIMER

K38
PASCAL RUNOFF WINNERS
FIRST - THIRD

K39
PASCAL RUNOFF WINNERS
FORTH & FIFTH

K40
PASCAL RUNOFF WINNERS
SIXTH PLACE

K41
EXPRESS 1.01 TEXT EDIT

K42
PASCAL RUNOFF-GRAPHICS

K43
PASCAL RUNOFF-GAMES

K44
PASCAL RUNOFF-PRINTERS

K45
PASCAL RUNOFF-UTILITIES

K46
PASCAL RUNOFF-TURBO UTILS

K47
256K RAM SOFTWARE

K48
C CONTEST WINNERS I

K49
C CONTEST WINNERS II

TCJ ***The Computer Journal***

P.O. Box 535, Lincoln, CA 95648-0535
Phone (916) 645-1670

Micro C Disks are $6.00 each plus shipping costs.

Shipping Cost to	U.S.	Canada/Mexico		Europe/Other	
		Surface	Air	Surface	Air
Added these costs	$1.00	$1.00	$1.25	$1.50	$2.50

Shipping costs are for GROUPS of 1 to 3 disks.

Computer Corner

By Bill Kibler

Regular Feature

Editorial Comment

60th Book Review

For this 60th issue, or Tenth Year special, I thought it would be appropriate to respond to several letters with one special Corner. Since taking over TCJ the most common request of new readers is what and where to find books on CP/M.

That idea prompted me to check my book shelves for CP/M books. Well I found more than just CP/M books and decided that my review should help all readers by commenting on some of my favorite books. The problem is availability, many are out of print. So to find some of these gems, you will need to search out swap meets or old book sales. In fact a few of these were found at swap meets for very little as well.

CP/M BOOKS

My recent purchase of a Kaypro for $50 included a full set of CP/M user manuals. They looked like new and contained most of the information a user needs. Since these were actually books by Digital Research Inc., any vendor of CP/M produced the same manuals with their own cover. So if you are looking for simple operational instructions, you usually can find a set or two at any computer swap meet that has old systems for sale (PC clone computer sales seldom have CP/M products or information - after all CP/M was in use before many of these people were born...).

For programming I found "The Programmer's CP/M Handbook" by Andy Johnson-Laird to be the best. This was an Osborne/McGraw-Hill publication (ISBN 0-88134-103-7) that I paid $21.95 for new in 1983. The best part of this book is the fact that it is written for people trying to put together their own system. It contains a sample BIOS and assembly code for writing to the BDOS. There is plenty of trouble shooting and system configuration help which should get first time hackers through that initial system port.

For those interested in the insides of CP/M or "how it really works", I suggest RP/M by Jack Denton of Oregon. This is a complete CP/M replacement that comes with source code. That is right, complete source of the entire system. Availability is in question at present, but since Jack is still a reader of *The Computer Journal,* I will be contacting him about getting it for our readers.

S-100 and Hardware

To be able to do hardware hacking on S-100 systems, requires some fundamental digital understanding. If you find your basic electronic skills a bit lacking you might look at "Electricity and Electronics" by Howard H. Gerrish and William E. Dugger from The Goodheart-Willcox Company (ISBN 0-87006-685-4). My college used several other books for our Introduction to Electronic course and finally settled on this one. It is fairly modern and gives a good overview of the electronic field. This is by no means a complete study but will get you started in the right direction. You might try your local Junior College book store, but be prepared for a $40 price tag.

For a cheaper introduction you might try the Sams Understanding Series and their "Understanding Electricity and Electronic Principles". These are by Training and Retraining Inc., a group that produces manuals for industrial training. This is Sams #27601 or ISBN 0-672-27061-7 and usually go for $18. There is a whole series of these introduction books, and in fact I like there Data Communications book for use as a desk reference. It contains many of the interface pinouts and standards described in a brief concise manner.

For those starting to get into S-100 you will need "The S-100 BUS Handbook" by Dave Bursky that was published by Hayden Book Company (ISBN 0-8104-0897-X). It cost $13 in 1980 and contains schematics of IMSAI power supply, CPU board, Front Panel logic, PROM board, UCRI, MIO, IFM, PIC-8. Also included are many Pertec boards, CPU, Front panel, memory, 2SIO, 4PIO, Disk Controller, AD/DA, and Process control interface. The book has more than schematics, there is a good intro to logic fundamentals, some review of circuits, interfaces, and programming concepts. This is a must have book for first time S-100 people. It should also be very hard to find, as most user will not give it up if still alive.

One intro book I believe still in print is the "Basic Microprocessors and the 6800" by Ron Bishop. For those who have little knowledge of how CPU's work, this book will answer it all. It uses the 6800 family of devices to explain how microprocessors talk and do their "stuff". It often has the Motorola name on it and can be found in their library (ISBN 0-8104-0758-2) or at any swap meet worth going to.

For interfacing to S-100 or any real system the best book is "Interfacing to S-100/IEEE 696 Microcomputers" by Sol Libes and Mark Garetz. This was still being published by M&T press, but when I called two months ago they said they no longer are stocking it. Originally it was an Osborne/McGraw-Hill book and

like the S-100 handbook most readers will hang on to this to the very end. It is worth whatever you have to pay to get one, but I am afraid it will only be found in libraries and estate sales.

Speciality Hardware

A book I found last year by Sybex is "Mastering Digital Device Control" by William G. Houghton. This $24.95 (ISBN 0-89588-346-5) book has plenty of examples and code for those interfacing 8048, 8051, or 6805 systems. There is a section on every type of interface problem you will encounter in designing and programming embedded systems. You will need to have at least a good understanding of how computer components work and fit together before reading this book, else some of the concepts may be a little hard to follow.

The old standby for getting fundamental logic information is the series of books by Don Lancaster. His series of work is still valid and covers TTL devices, CRT/TV typewriters, CMOS, RTL, Filters Design and most titles are appended with "COOKBOOK". Whichever one you get, you can expect a good book that covers the topic in detail and provides many sample schematics and pinouts.

FORTH

If your using any of the current Forth products, books by C.H. Ting are a must. He personally sells his books through Offete Enterprises, Inc. at 1306 South "B" Street, San Mateo, CA 94402 (415-574-8260). He has all the FPC manuals, as well as F83, Novix, and many more. Give him a call.

Also with a good set of publications is the Forth Interest Group listed on the inside cover of this issue. Every issue of their magazine/newsletter (called Forth Dimensions) has a listing of the current popular books and journals available from them on Forth. Joining FIG is a must do for any Forth programmer (tell them TCJ sent you.)

Another place with several books and a version of Forth popular for several years (and has no know bugs) is Mountain View Press. It just changed hands (back to the starter of MVP, Glen Haydon, and writer of some good books on the subject as well) and as such runs as a Division of Epsilon Lyra Inc. 19500 Skyline Blvd., Box 429, Route 2, La Honda, CA 94020, (415) 747-0760. Give Glen a call and he will send you his latest list of publications.

The best book on Forth is "Thinking Forth" by Leo Brodie who wrote the standard introduction "Starting Forth". This follow on book is not just on Forth but on programming in general. There are some really great topics presented in such a way that any programmer can see the advantages of changing the way they currently produce programs. Factoring is where you pull out sections of code that occur as small bits in many parts of your program. When factored properly, programs get smaller, faster and easier to maintain. Those concepts are all covered in this book (ISBN 0-13-917568-7). I recommend this book strongly.

For an inside look at Forth try "Threaded Interpretive Languages" by R.G. Loeliger (ISBN 0-07-038360-X). This book is available through FIG and covers a non-standard Z80 based forth system in assembly.

68000

For those working on 68000 systems I found a good book on doing assembly language that needs to be mentioned. The book is "68000 Assembly Language Programming" by Gerry Kane, Doug Hawkins, and Lance Leventhal. What I like about this book is their completeness on the subject. They explain how to do assembly, tips and hints, tricks, and what it is you are doing to begin with. A good beginners book that has samples and covers the support chips as well. It is by McGraw-Hill as ISBN 0-07-931062-1 and I have the first printing in 1981, but I believe there is a more current printing available.

FUN!

Lastly are three books of interest and fun thinking. By fun thinking I mean a book that tickles the brain into doing things differently. I attended a meeting many years ago in which the guest speaker was Roger von Oech. It was fun and great for getting you going into many new directions. His books "A whack on the side of the Head" and "A Kick in the Seat of the Pants" are just great. He says "The human body has two ends on it: one to create with and one to sit on. Sometimes people get their ends reversed." These are must read books if you do anything creative at all.

My last book has a personal aspect to it. My old high school buddy is Joseph Killian, who if you remember your IMSAI history designed the first IMSAI 8080 systems. I worked with Joe for awhile at MicroPro (the WordStar people) when they were building the MBP 1000 (I drew the schematics and supported the designer.) While there I learned that Joe was suing Bill Millard the starter of IMSAI and owner of Computerland for stock in Computerland. Seems Millard gave Joe interest in IMSAI as payment for his work in making the original IMSAI. But what happened was that Millard then started Computerland as the suit says and wasted IMSAI and made the stock worthless in the process.

Last month I saw the book "Once Upon A time In Computerland" by Johanthan Littman (ISBN 0-671-69392-1), it was a $14.95 paperback on sale for $2.88. It had Joe's picture in it as well as telling the story of the court cases involved. So I picked it up to find out what happened to my friend. The surprise was the quality of the book. It reads like a best selling "who done it", only you have to remind yourself regularly that these are true stories based on court records and depositions. If you have ever wondered what went on in those early days of computing, here it is. In fact it is still going on as of 1990 when all sides were going to appeal the court results. It is great reading and a must for anybody who wants to know about EST and Computerland.

That's All...

Well those are my selections on books, how about yours? Till next time.. . Bill.

Discover
The Z-Letter
The Z-letter is the only monthly publication for CP/M and Z-System. Eagle computers and Spellbinder support. Licensed CP/M distributor.

Subscriptions: $18 US, $22 Canada and Mexico, $36 Overseas. Write or call for free sample.

The Z-Letter
Lambda Software Publishing
149 West Hilliard Lane
Eugene, OR 97404-3057
(503) 688-3563

Advent Kaypro Upgrades
TurboROM. Allows flexible configuration of your entire system, read/write additional formats and more, only $35.
Personality Decoder Boards
Run more than two drives when using TurboROM, $25.
Hard Drive Conversion Kits. Call or write for availability & pricing.

Call (916)483-0312
eves, weekends or write
Chuck Stafford
4000 Norris Ave.
Sacramento, CA 95821

TCJ MARKET PLACE
Advertising for small business
First Insertion: $50
Reinsertion: $35

Rates include typesetting. Payment must accompany order. VISA, MasterCard, Discover, Diner's Club, Carte Blanche, JCB, EuroCard accepted. Checks, money orders must be US funds. Resetting of ad consitutes a new advertisement at first time insertion rates.
Mail ad or contact ,
The Computer Journal
P.O. Box 535
Lincoln, CA 95648-0535

CP/M SOFTWARE

100 page Public Domain Catalog, $8.50 plus $1.50 shipping and handling. New Digital Research CP/M 2.2 manual, $19.95 plus $3.00 shipping and handling. Also, MS/PC-DOS Software. Disk Copying, including AMSTRAD. Send self addressed, stamped envelope for free Flyer, Catalog $1.00

Elliam Associates
Box 2664
Atascadero, CA 93423
805-466-8440

8 BITS *and Change*

CLOSING OUT SALE!

All 12 Back Issues

for only $40

Send check to

Lee Bradley
24 East Cedar Street
Newington, CT 06111
(203) 666-3139 voice
(203) 665-1100 modem

S-100/IEEE-696

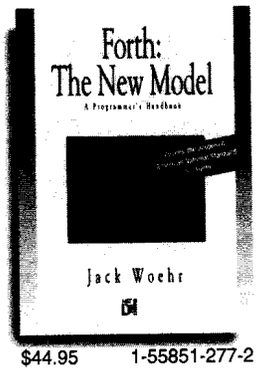
Compupro
Cromemco
IMSAI
and more!

Cards • Docs • Systems

Dr. S-100

Herb Johnson,
CN 5256 #105,
Princeton, NJ 08543
(609) 588-5316

New from M&T Books!



$44.95 1-55851-277-2

M&T BOOKS
Technical Books for Technical Times

Available at bookstores everywhere
or call 1-800-688-3987
RCJ3

Z80 STD USERS!

Cost Effective Upgrade
Clock Speeds to 10 MHz
1 Mbyte On-board Memory

Increase your system performance and reliability while reducing your costs by replacing three of the existing cards in your system with one **Superintegrated** Z80 Card from Zwick Systems.

A **Superintegrated** Card in your system protects your software investment, requiring only minor changes to your mature Z80 code. You can increase your processing performance by up to 300 percent in a matter of days!

Approximatly 35 percent of each **Superintegrated** Card has been reserved for custom I/O functions including A/D, D/A, Industrial I/O, Parallel Ports, Serial Ports, Fax and Data Modems or almost any other form of I/O that you are currently using.

Call or Fax today for complete information on this exciting new line of **Superintegrated** Cards and upgrade your system the easy way!

ZWICK SYSTEMS INC.
Tel (613) 726-1377, Fax (613) 726-1902